UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jan Tomšič

Podatkovna shramba kot spletna storitev

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jan Tomšič

Podatkovna shramba kot spletna storitev

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: dr. Andrej Brodnik

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pojem računalnika se je povsem spremenil. Danes dejansko nimamo več računalnika, ki bi imel enoto za hranjenje podatkov, enoto za obdelavo podatkov, enoto za prikazovanje podatkov in tako naprej. Naša obdelava podatkov se dogaja nekje – v oblaku.

V tej diplomski nalogi se osredotočite na shranjevanje podatkov. Preglejte najpogostejše sisteme za hranjenje datotečnih podatkov in za hranjenje predmetov (SaaS). Med slednjimi se osredotočite na sistem CEPH, ki omogoča zasebni SaaS. Primerjajte vse sisteme iz zornega kota uporabnosti, hitrosti in robustnosti.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Jan Tomšič sem avtor diplomskega dela z naslovom:

*Podatkovna shramba kot spletna storitev (*angl. *Data storage as an online service)*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom dr. Andreja Brodnika
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 6. marca 2016 Podpis avtorja:

Zahvaljujem se mentorju dr. Andreju Brodniku za usmerjanje in napotke pri izdelavi diplomske naloge in Andreju Toliču za napotke in literaturo. Zahvaljujem se še sodelavcu Metodu Novaku za infrastrukturo, ki je omogočila praktičen del te naloge. Predvsem pa sem hvaležen staršem, ki so me podpirali skozi celoten proces šolanja, ter svoji dragi Margot in sinu Aleksandru za ljubezen, moralno in čustveno podporo ob izdelavi diplomske naloge.

Kazalo

Kazalo slik

Kazalo izpisov

Kazalo tabel

Seznam uporabljenih kratic

kratica	angleško	slovensko
APT	*Advanced Packaging Tool*	orodje za upravljanje s paketi v operacijskih sistemih Debian
CIFS	*Common Internet File System*	splošni spletni datotečni sistem
FSID	*File Space Identifier*	oznaka datotečnega sistema
FQDN	*Fully Qualified Domain Mame*	popolno domensko ime
FUSE	*Filesystem in Userspace*	datotečni sistem v uporabniškem prostoru
MDS	*Metadata Server*	strežnik za meta podatke
NFS	*Network File System*	omrežni datotečni sistem
NTP	*Network Time Protocol*	omrežni časovni protokol
OSD	*Object Storage Device*	naprava za shranjevanje predmetov
PG	*Placement Group*	postavitvena skupina
POSIX	*Portable Operating System Interface*	prenosni vmesnik operacijskega sistema
RADOS	*Reliable Autonomic Distributed Object Store*	zanesljiva, avtonomna, porazdeljena shramba predmetov
RBD	*RADOS Block Device*	RADOS bločna naprava
REST	*Representational State Transfer*	predstavitveni prenos stanja
RPC	*Remote Procedure Call*	klic oddaljenega postopka
S3	*Simple Storage Service*	storitev enostavne shrambe
SMB	*Server Message Block*	blok s sporočilom strežnika
SSH	*Secure Shell*	varna lupina
SSL	*Secure Sockets Layer*	sloj varnih vtičnic

URI	*Uniform Resource Identifier*	enotni označevalnik vira
VFS	*Virtual File System*	navidezni datotečni sistem

Razlaga uporabljenih tujih izrazov

izraz	**angleško**	**pomen v slovenščini**
entiteta	*entity*	Kar obstaja, ima identiteto in natančen pomen.
konfiguracija	*configuration*	Nastavitve ali datoteka z nastavitvami.
metoda	*method*	Postopki in pravila za izvajanje določene naloge.
monitor	*monitor*	Programska oprema, ki nadzoruje dostop do podatkov in naprav v informacijskem sistemu.
protokol	*protocol*	Predpisano zaporedje postopkov za izvajanje.
repozitorij	*repository*	Okolje, kjer je po določenih kriterijih zbrano, urejeno in shranjeno elektronsko gradivo, podatki, npr. dokumentacija in programski paketi.

Povzetek

Namen diplomskega dela je primerjava datotečnih sistemov in predmetne shrambe z dostopom preko računalniškega omrežja. Predstavili smo omrežni datotečni sistem in priključitev shrambe v operacijskem sistemu Linux. Opisali smo predmetno shrambo in način shranjevanja podatkov v porazdeljeni shrambi. Predstavljen je vmesnik REST za dostop do predmetne shrambe Amazon S3. Podrobno je opisan sistem za porazdeljeno predmetno shrambo Ceph, ki smo ga za namen diplomske naloge postavili na gruči strežnikov. Opisana je shramba RADOS, na kateri je zasnovan Ceph, in datotečni sistem CephFS, ki nudi datotečni sistem, skladen s POSIX. Opravili smo meritve hitrosti in porabe procesorskega časa prenosa podatkov med shrambami z uporabo različnih vmesnikov in protokolov z rezultati, prikazanimi z grafikoni.

Ključne besede: shranjevanje podatkov, datotečni sistem, shramba predmetov, Ceph, Amazon S3, spletna storitev

Abstract

The purpose of the thesis was comparison of interfaces to network attached file systems and object storage. The thesis describes network file system and mounting procedure in Linux operating system. Object storage and distributed storage systems are explained with examples of usage. Amazon S3 is an example of object store with access trough REST interface. Ceph, a system for distributed object storage, is explained in detail, and a Ceph cluster was deployed for the purpose of this thesis. Ceph is based on object store RADOS and offers POSIX compliant file system CephFS. We measured data transfer rates and processing time of file system and object store using various protocols and interfaces. The results of measurements are displayed in graphs.

Keywords: data storage, file system, object storage, Ceph, Amazon S3, online service

Poglavje 1 Uvod

Količina podatkov, ki jih dnevno ustvarimo, raste vsak dan. Podatke želimo shraniti v zanesljivi shrambi, ki jo po potrebi lahko razširimo. Do shrambe želimo dostopati kjerkoli in kadarkoli, kar nam omogoča svetovni splet. Za temo diplomske naloge smo opisali shrambo podatkov kot spletno storitev. Primerjali bomo datotečni sistem in predmetno shrambo po hitrosti prenosa in procesorski učinkovitosti. Do obeh tipov shrambe bomo dostopali preko različnih protokolov preko spleta ali zasebnega omrežja.

Opisali bomo porazdeljeno shrambo, katere prednosti so predvsem v zanesljivosti in enostavni razširljivosti. Predstavili bomo datotečni sistem, ki nudi hiter prenos podatkov preko omrežja, a je slabo razširljiv. Osredotočili se bomo predvsem na predmetno shrambo in prednosti pred datotečnim sistemom. Predmetna shramba omogoča shranjevanje izjemno velikih količin podatkov, je zanesljiva, enostavno razširljiva in preprosta za uporabo. Opisali bomo vmesnik REST, s katerim do predmetne shrambe dostopamo preko spleta. Kot primer predmetne shrambe bomo navedli Amazon S3, kot primer porazdeljenega datotečnega sistema in predmetne shrambe pa brezplačen sistem Ceph, ki ga bomo namestili na lastni strojni opremi.

1.1 Struktura naloge

Po kratkem uvodu bomo v drugem poglavju opisali predmetno shrambo in predstavili Amazon S3 ter dostop do shrambe preko vmesnika REST. Definirali bomo predmet in opisali prednosti predmetne shrambe pred datotečno in bločno shrambo podatkov.

V tretjem poglavju bomo opisali datotečni sistem. Podrobneje bomo opisali protokol NFS in omenili Sambo. Prikazali bomo še način priključitve datotečnega sistema v uporabniškem okolju (FUSE), ter na ta način priključili predmetno shrambo S3.

V poglavju 4 bomo opisali sistem za porazdeljeno, predmetno shrambo Ceph. Predstavili bomo arhitekturo in vse elemente sistema ter način, na katerega Ceph shranjuje in vrača podatke. Opisali bomo še datotečni sistem CephFS in pokazali način priključitve takega datotečnega sistema na odjemalce.

Za konec bomo postavili sistem Ceph in ga primerjali z Amazonovo shrambo S3. V poglavju 5 bomo obe shrambi primerjali po hitrost nalaganja podatkov na strežnik in snemanja podatkov s strežnika preko vmesnika REST. Poleg predmetne shrambe bomo po hitrosti primerjali še datotečne sisteme CephFS, NFS, Sambo in s3fs.

V dodatkih A in B bomo opisali namestitev strežnika in odjemalca NFS in Samba. V dodatku C bomo podrobno opisali postavitev sistema Ceph na gručo strežnikov, namestitev prehoda RADOS in uporabo datotečnega sistema CephFS.

Poglavje 2 Predmetna shramba

Predmetna shramba je način shranjevanja podatkov, ki naslavlja pomanjkljivosti razširljivosti in varnosti bločne shrambe in njenih vmesnikov. Najmanjša enota v taki shrambi je predmet (angl. *object*), ki je logična zbirka bajtov na pomnilniški napravi in je za razliko od blokov lahko raznolikih velikosti [1]. Vsak predmet sestoji iz podatkov samih, shranjenih v binarni obliki, enoličnega identifikatorja in meta podatkov, ki so lahko poljubni, saj z njimi upravlja aplikacija oziroma odjemalec. Meta podatki običajno vsebujejo lastnosti kot so ime, tip shranjenih podatkov, čas zadnjega dostopa, pravice in ključi za dostop. Odjemalec ali aplikacija lahko za potrebe svojega delovanja shrani poljubne meta podatke predmetov.

Predmetna shramba nudi dobre lastnosti tako datotečne, kot bločne shrambe. Datotečni dostop aplikaciji nudi višji nivo abstrakcije nad shranjevanjem s čimer omogočijo varno souporabo datotek med različnimi operacijskimi sistemi, a večinoma na račun slabše zmogljivosti zaradi preobremenjenosti datotečnega strežnika. Bločni dostop nudi hiter, neposreden dostop do blokov, vendar brez strežnika za overjanje dostopa in vzdrževanja meta podatkov, za kar mora skrbeti aplikacija oziroma odjemalec sam. Shramba predmetov nudi prednosti tako datotek kot blokov. Podobno blokom, so predmeti osnovni podatkovni tip in jih lahko naslavljamo neposredno, brez potrebe po vmesnem strežniku, s čimer dosežemo zelo dobre zmogljivosti in hitrost predmetne shrambe. Primerljivo z dostopom do datotek, odjemalci do predmetov dostopajo preko vmesnika, ki aplikacijo loči od meta podatkov, potrebnih za postopek shranjevanja predmeta, kar zagotovi enostavno dostopnost iz kateregakoli okolja, bodisi so to različni operacijski sistemi ali lastne aplikacije [2].

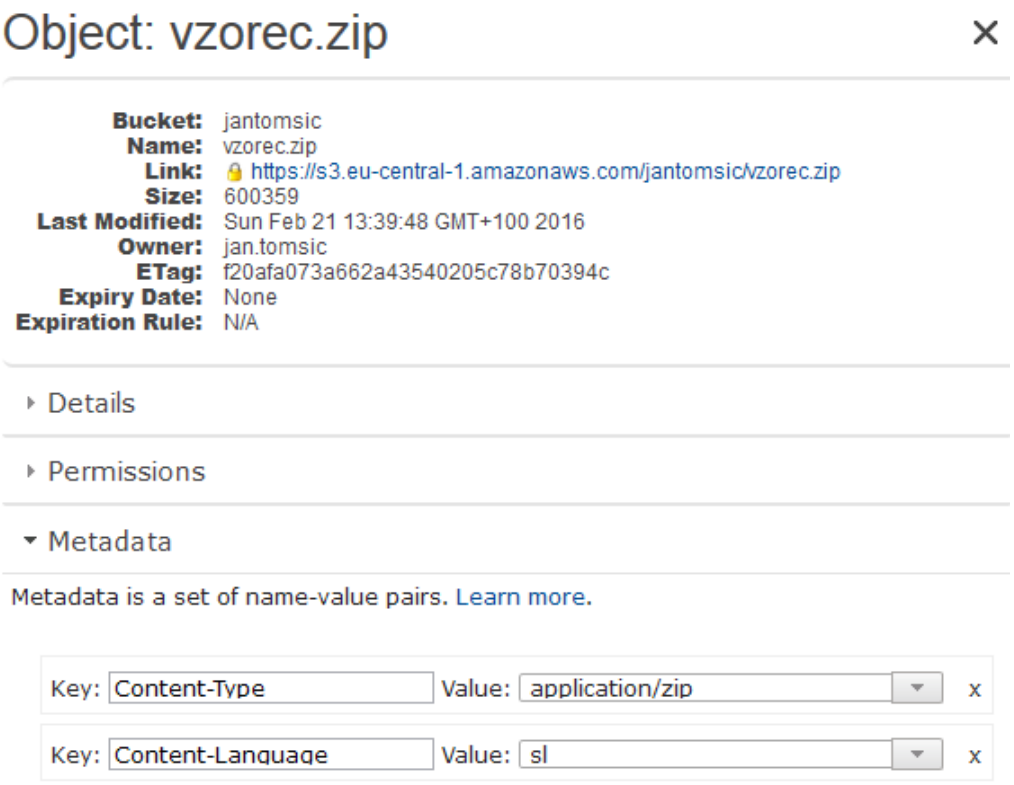
2.1 Amazon S3

Amazon S3 (*Simple Storage Service*) [3] je komercialna storitev, ki ponuja shrambo v oblaku, in je del ponudbe spletnih storitev Amazon Webservices. Shrambo S3 bomo opisali kot primer predmetne shrambe in predstavili aplikacijski vmesnik REST, s katerim do takšne shrambe dostopamo.

Vedro je osnovna logična pomnilniška enota, v kateri so predmeti shranjeni. Namen veder je identifikacija računa za namen zaračunavanja porabljenega prostora in prenesenih podatkov.

Nad vedri lahko izvajamo nadzor dostopa do podatkov le določenim uporabnikom. Nekatere možnosti, ki jih S3 ponuja, kot je na primer upravljanje z verzijami, lahko nastavljamo le na nivoju celotnega vedra.

Predmet je temeljna entiteta shrambe S3. Vsak predmet sestoji iz podatkov samih in meta podatkov, ki opisujejo predmet z vrednostmi, predstavljenimi kot pari ključev in vrednosti. Vsak predmet vsebuje nekaj privzetih meta podatkov, kot so ime in velikost predmeta, vedro, v katerem je predmet shranjen in čas zadnje spremembe. Ostali meta podatki, kot na primer tip vsebine, so zapisani v standardni obliki, ki se uporablja v glavi parametrov pri transakcijah, opravljenih preko protokola HTTP. Slednji so namenjeni le aplikaciji, ki do teh predmetov dostopa, in ne vplivajo shranjeno vsebino. Slika 2.1 prikazuje lastnosti predmeta vzorec.zip. Iz slike lahko razberemo privzete meta podatke, ki so označeni odebeljeno, ter lastne meta podatke, ki se nahajajo v spodnjem delu.



Slika 2.1: Lastnosti predmeta vzorec.zip, shranjenega v vedru jantomsic.

Ključ (angl. *key*) enolično identificira vsak predmet v vedru. V shrambi S3 so predmeti enolično naslovljivi s kombinacijo imena vedra, ključa in različice, v vedrih kjer je možnost upravljanja z različicami omogočena. Amazon S3 si lahko zamislimo kot slovar, ki preslika vrednost

»vedro + ključ + različica« v predmet. S to vrednostjo lahko predmet neposredno naslovimo, na primer preko naslova URI.

Upravljanje z različicami v S3 omogoča hranjenje več inačic istega predmeta v vedru. Z hranjenjem starejših različic se lahko obvarujemo pred izgubo podatkov v primeru, da predmet prepišemo ali izbrišemo, saj se ob vsaki spremembi predmeta le ta shrani kot nov predmet z istim ključem in novo oznako verzije (angl. *version ID*) [4]. Slika 2.2 prikazuje seznam predmetov in njihovih različic. Vsaka različica je nov predmet, ki ga naslavljamo s kombinacijo imena in oznako verzije.

	Name / Version Create Date	Storage Class	Version ID
	album.zip	--	--
	Wed Feb 10 20:39:15 GMT+100 2…	Standard	eGiuyXBfG9xHvkLC0u4uCeXC7UcWeFpi
	Wed Feb 10 20:25:56 GMT+100 2…	Standard	Wkkb1ZSQhD9b386SB1MbLdyrWutmVto5
	backup.zip	--	--
	Wed Feb 10 20:40:25 GMT+100 2…	Standard	9cWahgR0_1QIXXIeA.4E1I9bfaRIEgxe
	Wed Feb 10 20:39:13 GMT+100 2…	Standard	qLRDM0Yufsc3McGI.mPIc1.z4u.Jvy4P
	Tue Feb 09 23:11:10 GMT+100 2016	Standard	alijR39xFODpuLyGe1CsR0_XC08daklu
	inet99.pdf	--	--
	Mon Feb 08 16:58:51 GMT+100 2…	Standard	d_5P2IP2QemLbKPGUZ1LSJP5LGYD2CnL
	paper.pdf	--	--
	Mon Feb 08 16:58:56 GMT+100 2…	Standard	INM_a23SC81D0HiWM75LoOuPv3CEs36n

Slika 2.2: Pregled starejših različic predmetov v vedru z omogočenim upravljanjem z različicami.

2.1.1 Programski vmesnik REST

Za dostop do podatkov nudi S3 vmesnik REST preko protokola HTTPS. REST je implementacija programerske arhitekture namenjene učinkovitejši komunikaciji med računalniškimi sistemi, primerna predvsem za skupno rabo velikih količin podatkov. Ideja vmesnika je podatke nuditi na zahtevo in z ostalimi deliti le reference na podatke namesto celotne kopije podatkov samih [5]. V primeru svetovnega spleta, ki je največji primer sistema, ki temelji na REST, spletne vire naslavljamo z naslovom URI.

Za dostop do S3 preko vmesnika REST najprej potrebujemo ključ za dostop in skrivni ključ, ki jih generiramo v nadzorni plošči Amazonovih spletnih storitev. Z vmesnikom komuniciramo z uporabo standardnih metod (GET, HEAD, POST, PUT in DELETE), ki so definirane v standardu HTTP/1.1. S temi metodami lahko izpišemo vsebino vedra, nalagamo in brišemo predmete in upravljamo z drugimi nastavitvami, ki jih S3 ponuja.

Za izvajanje poizvedb z vmesnikom shrambe S3 smo uporabili knjižnico boto [6] in programski jezik Python. Z metodo GET izpišemo vsebino vedra (Izpis 2.1). Metodi kot argumente podamo ime vedra, naslov gostitelja, avtorizacijski ključ in datum poizvedbe. Rezultat je seznam ključev, ki jih potrebujemo za dostop do predmetov v tem vedru.

```
GET /jantomsic/ HTTP/1.1
Accept-Encoding: identity
Host: s3.eu-central-1.amazonaws.com:443
x-amz-content-sha256:
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
X-Amz-Date: 20160222T091909Z
Authorization: AWS4-HMAC-SHA256
Credential=FRIAJTXN6J26XAOZJANQ/20160222/eu-central-1/s3/aws4_request,
SignedHeaders=host;user-agent;x-amz-content-sha256;x-amz-date,Signatur
e=ec8244c361d5ec36d27689f99e8928f7d968c3045748ab854e13970801302d4a
User-Agent: Boto/2.38.0 Python/3.5.1 Windows/8.1
Content-Length: 0

HTTP/1.1 200 OK

<Key: jantomsic,Primerjava NFS in SMB.docx>
<Key: jantomsic,Turn_On_Taskbar_Animations.reg>
<Key: jantomsic,dokumenti/>
<Key: jantomsic,dokumenti/Diploma-jant-hp.docx>
<Key: jantomsic,objekt.vsdx>
<Key: jantomsic,paper.pdf>
<Key: jantomsic,paxos.txt>
<Key: jantomsic,s3fs ls.png>
<Key: jantomsic,vzorec.zip>
<Key: jantomsic,zabbix.png>
```

Izpis 2.1: Izpis vsebine vedra z metodo GET, ki vrne seznam ključev v vedru.

Za nalaganje podatkov v vedro uporabimo metodo PUT, katere uporabo prikazuje Izpis 2.2. Metodi v naslovu URI podamo ključ do predmeta in ostale vrednosti, ki opisujejo predmet in vsebujejo velikost, tip vsebine in kontrolno vsoto, ki jo odjemalec izračuna pred in po prenosu datoteke s strežnika. Metoda PUT vrne le kodo odziva po standardu HTTP/1.1, ki je v primeru uspešne izvedbe `200 OK`. Predmete iz shrambe izbrišemo z metodo DELETE. Naslov URI je pot to ključa, ki ga želimo izbrisati. Ob uspešnem izbrisu metoda vrne kodo odziva `204 No Content`. Za izbris predmeta »backup.zip« iz vedra »jantomsic« uporabimo ukaz `DELETE /jantomsic/backup.zip` (Izpis 2.3). V glavi poizvedbe podamo podatke za avtorizacijo in ostale, kot so prikazani v izpisih Izpis 2.1 in Izpis 2.2.

```
PUT /jantomsic/scan0002.jpg HTTP/1.1
Accept-Encoding: identity
Content-Length: 155687
Content-MD5: 3rk4tzKqADx40TY2wh4R/w==
Authorization: AWS4-HMAC-SHA256
Credential=FRIAJTXN6J26XAOZJANQ/20160222/eu-central-1/s3/aws4_request,
SignedHeaders=content-length;content-md5;content-type;expect;host;user
-agent;x-amz-content-sha256;x-amz-date,Signature=86aa4697dacb63ea046e7
96b510869fe95da5fc1b45a3b4374c9747c63c3d9a9
X-Amz-Date: 20160222T095834Z
Content-Type: image/jpeg
Host: s3.eu-central-1.amazonaws.com:443
Expect: 100-Continue

HTTP/1.1 100 Continue
HTTP/1.1 200 OK
```

Izpis 2.2: Nalaganje datoteke shrambo S3 z metodo PUT.

```
DELETE /jantomsic/scan0001.jpg HTTP/1.1
Accept-Encoding: identity
Content-Length: 0
User-Agent: Boto/2.38.0 Python/3.5.1 Windows/8.1
Host: s3.eu-central-1.amazonaws.com:443

reply: 'HTTP/1.1 204 No Content'
```

Izpis 2.3: Izbris predmeta iz shrambe S3 z medoto DELETE.

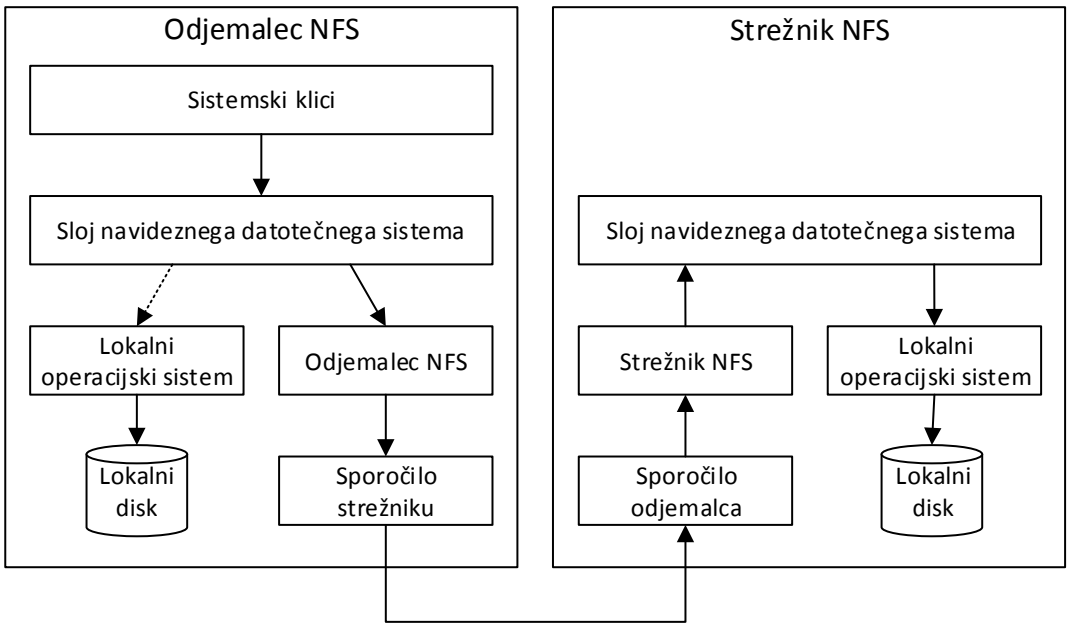
Poglavje 3 Datotečni sistem

V tem poglavju bomo opisali dostop do omrežnih datotečnih sistemov. Datotečni sistemi, do katerih dostopamo preko omrežja, nudijo sistemu enak vmesnik kot lokalen datotečni sistem. Opisali bomo protokola NFS in Sambo ter uporabo predmetne shrambe S3 kot datotečni sistem v uporabniškem načinu.

3.1 Omrežni datotečni sistem NFS

Omrežni datotečni sistem NFS je razvilo podjetje Sun leta 1984. Ideja protokola NFS je neodvisnost od strojne opreme in operacijskega sistema, možnost obnovitve po sesutju odjemalca in zadovoljiva zmogljivost in hitrost prenosa. Kljub starosti je NFS še danes v razvoju, saj je en izmed najpogosteje uporabljanih omrežnih datotečnih sistemov.

NFS je protokol, ki odjemalcem nudi neposreden dostop do datotečnega sistema na oddaljenem strežniku, dosegljivem preko omrežja. Odjemalec, ki se priključi na strežnik NFS, upravlja z datotekami na strežniku enako, kot bi upravljal z lokalnimi datotekami, saj je NFS popolnoma skladen s POSIX. Sloj navideznega datotečnega sistema (VFS, angl. *virtual file system*) v operacijskem sistemu Linux omogoča aplikacijam komunikacijo z omrežnimi datotečnimi sistemi na enak način kot z lokalnimi. VFS predpisuje vmesnik med jedrom in datotečnim sistemom. Slika 3.1 opisuje delovanje protokola NFS, ki je implementiran na podlagi mehanizma klicev oddaljenega postopka (RPC, angl. *remote procedure call*). Klic oddaljenega postopka pomeni klic vnaprej določenega dela programa, ki se nahaja na drugem strežniku. Vsak klic postopka je v NFS sinhron, kar pomeni, da mora odjemalec pred vsakim naslednjim klicem počakati na rezultat prejšnjega. To preprečuje, da bi odjemalec prejel stare podatke ali da bi star klic prepisal podatke novejšega klica. NFS je protokol brez pomnjenja predhodnih akcij (angl. *stateless protocol*), kar pomeni da mora odjemalec strežniku NFS ob vsakem klicu posredovati vse potrebne parametre potrebne za uspešen klic. Če je klic neuspešen, odjemalec ponavlja klice dokler strežnik ne vrne ustreznega rezultata. Zgodnje verzije protokola NFS niso podpirale možnosti overjanja dostopa ali šifriranja prometa. Edini način omejevanja dostopa je seznam odjemalcev, ki jim dovolimo dostop do določenih datotečnih sistemov. Odjemalci so navedeni z imenom (angl. *hostname*) ali naslovom IP. Overjanje dostopa in šifriranje povezave podpira novejša verzija protokola NFS4 [7] [8].



Slika 3.1: Za vsak sistemski klic do datotečnega sistema NFS odjemalec pošlje sporočilo strežniku NFS. Strežnik iz sporočila razbere ukaz, ga izvede, in rezultat po isti poti vrne odjemalcu.

3.2 Samba

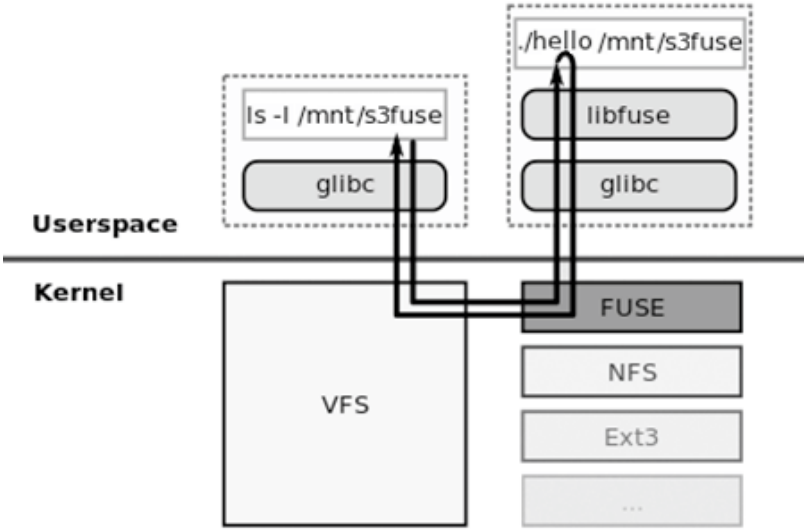
Samba je odprtokodna implementacija protokola SMB/CIFS, ki operacijskemu sistemu Linux nudi souporabo datotek s sistemom Windows in drugimi operacijskimi sistemi. Protokol SMB je protokol zahteve in odgovora (angl. *request-response*), kar pomeni, da odjemalec pošlje sporočilo SMB strežniku, ki mu s sporočilom SMB odgovori. Sporočilo SMB je sestavljeno iz glave enotne velikosti in ukaznega niza, ki je lahko različne velikosti [9].

Protokol SMB nudi overjanje dostopa z uporabniškimi imeni in gesli ter pravicami nad datotekami. Odjemalec za SMB je prisoten v večini modernih operacijskih sistemih, zaradi česar je SMB eden izmed najbolj razširjenih protokolov za souporabo datotek preko omrežja.

3.3 Predmetna shramba kot datotečni sistem v uporabniškem načinu

Shrambo S3 lahko priključimo v operacijski sistem Linux s programom s3fs [10], ki nudi uporabo datotečnega sistema v uporabniškem načinu preko gonilnika FUSE v jedru lokalnega operacijskega sistema (Slika 3.2). Vsaka poizvedba v takem sistemu se preko gonilnika FUSE posreduje aplikaciji v uporabniškem načinu, ki poizvedbo obdela in po isti poti vrne rezultat. Gonilnik FUSE je v strukturi datotečnih sistemov na enakem nivoju kot odjemalec NFS na sliki

Slika 3.1, torej pod slojem navideznega datotečnega sistema. Aplikacije lahko ta gonilnik izkoristijo za priklop datotečnih sistemov, ki sicer niso skladni s standardom POSIX.



Slika 3.2: Priklop datotečnega sistema v uporabniškem načinu [11].

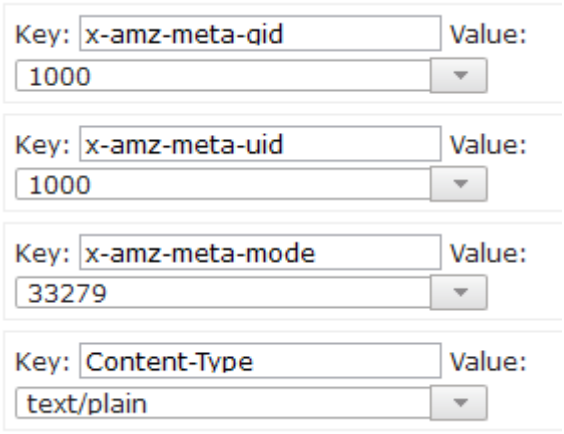
Z ukazom s3fs preko priključne točke `s3` pridobimo možnost dostopa do predmetov, shranjenih v vedru `jantomsic` (Izpis 3.1). Premeti so predstavljeni kot datoteke, nad katerimi lahko izvajamo običajne operacije operacijskega sistema Linux, kot so kopiranje, premikanje, brisanje in spreminjanje pravic dostopa. Z ukazom `ls` izpišemo vsebino imenika (Izpis 3.2) in opazimo, da je lastništvo in pravice datotek nastavljeno enako kot pri kateri koli drugi datoteki v običajnih datotečnih sistemih, ki jih Linux uporablja (npr. ext4). Uporabniške pravice in lastništvo datotek se shranijo kot meta podatki predmeta (Slika 3.3). Meta podatek x-amz-meta-gid in x-amz-meta-uid predstavljata ID skupine in uporabnika, ki jima datoteka pripada, x-amz-meta-mode pa bralno pisalne pravice zapisane v desetiški obliki.

```
jan@itpc:~/mounting_scripts$ s3fs jantomsic /mnt/s3 -o passwd_file=.s3access
FUSE library version: 2.9.4
nullpath_ok: 0
nopath: 0
utime_omit_ok: 0
unique: 1, opcode: INIT (26), nodeid: 0, insize: 56, pid: 0
INIT: 7.23
flags=0x0003f7fb
max_readahead=0x00020000
   INIT: 7.19
   flags=0x00000019
   max_readahead=0x00020000
   max_write=0x00020000
   max_background=0
   congestion_threshold=0
   unique: 1, success, outsize: 40
```

Izpis 3.1: Ukaz za priključitev vedra shrambe S3 kot datotečni sistem s3fs in izpis ob uspešni priključitvi z vključeno možnostjo razhroščevanja.

```
root@pita2:/home/pi/amazons3/s3# ls -lh
total 4.9M
-rwxrwxrwx 1 pi pi 598K Feb 22 14:40 backup.zip
drwxrwxrwx 1 pi pi    0 Feb  8 17:02 dokumenti
-rwxrwxrwx 1 pi pi  25K Feb 22 14:40 objekt.vsdx
-rwxrwxrwx 1 pi pi 227K Feb 22 14:40 osdi2002.pdf
-rwxrwxrwx 1 pi pi 178K Feb 22 14:40 paper.pdf
-rwxrwxrwx 1 pi pi  819 Feb 22 14:40 paxos.txt
-rwxrwxrwx 1 pi pi  19K Feb 22 14:40 Primerjava NFS in SMB.docx
-rwxrwxrwx 1 pi pi 1.1M Feb 22 14:40 s3fs ls.png
-rwxrwxrwx 1 pi pi  648 Feb 22 14:40 Turn_On_Taskbar_Animations.reg
-rwxrwxrwx 1 pi pi 587K Feb 22 14:40 vzorec.zip
-rwxrwxrwx 1 pi pi 2.2M Feb 22 14:40 zabbix.png
```

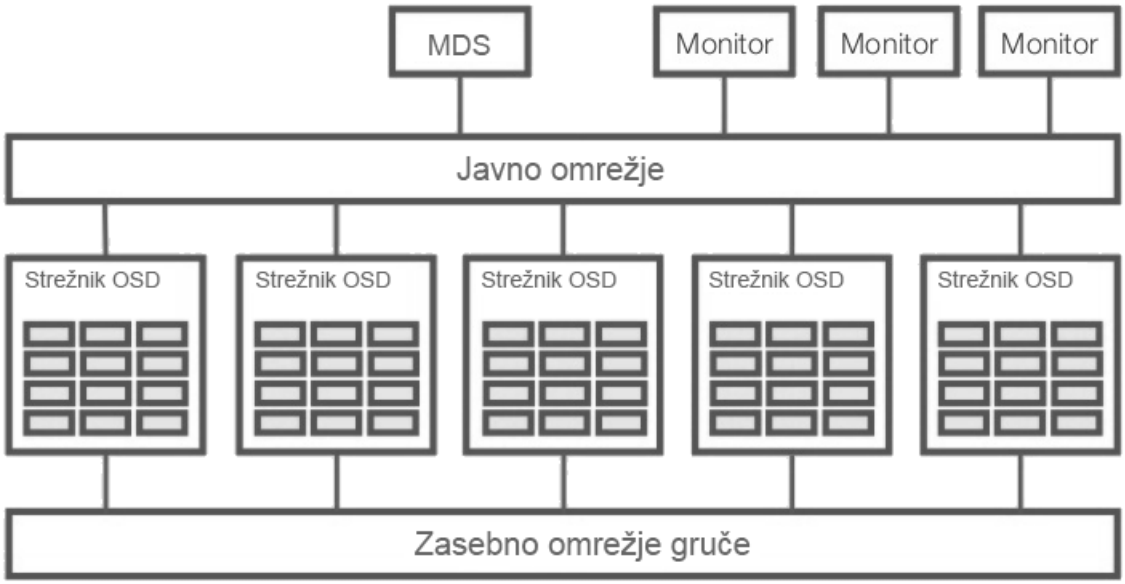
Izpis 3.2: Izpis vsebine imenika z ukazom `ls`.



Slika 3.3: Uporabniške pravice in lastništvo nad podatki so shranjeni kot meta podatki predmeta. Ključa *gid* in *uid* predstavljata ID skupine in uporabnika, katerima datoteka pripada na operacijskem sistemu Linux. Ključ *mode* shranjuje bralno pisalne pravice shranjene v desetiški obliki.

Poglavje 4 Sistem za porazdeljeno shrambo Ceph

V tem poglavju bomo predstavili sistem Ceph, sistem za porazdeljeno shranjevanje podatkov. Ceph odjemalcem nudi predmetno, datotečno in bločno shrambo. Vsi podatki, vključno z datotekami, bloki in meta podatki so shranjeni kot predmeti v porazdeljeni, predmetni shrambi RADOS, ki podatke shranjuje na gručo predmetnih shranjevalnih naprav OSD. Ceph nudi datotečni sistem CephFS, do katerega lahko dostopamo z običajnimi POSIX ukazi. Z meta podatki datotečnega sistema upravlja gruča strežnikov za meta podatke MDS. Slika 4.1 prikazuje arhitekturo sistema Ceph. Ceph sestavljajo strežniki OSD, ki vsebujejo pomnilniške medije, monitorji, ki nadzorujejo stanje gruče in strežniki za meta podatke, ki skrbijo za operacije v datotečnem sistemu. Množica OSDjev je med seboj povezana s hitrim zasebnim omrežjem za potrebe prenosa podatkov. Namestitev sistema Ceph je opisana v dodatku C.



Slika 4.1: Arhitektura sistema Ceph.

4.1 Porazdeljena shramba predmetov RADOS

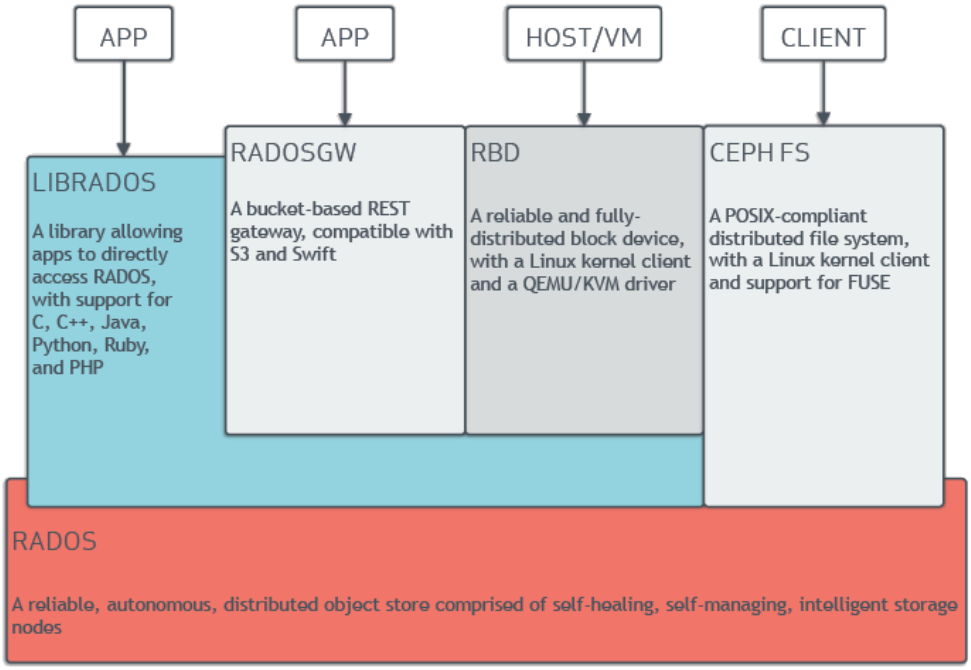
Zanesljiva, avtonomna, porazdeljena shramba predmetov (angl. *Reliable Autonomic Distributed Object Store – RADOS*) je shranjevalni sistem, na katerem temelji Ceph. RADOS skrbi za migracijo in replikacijo podatkov ter zaznavanje in odpravljanje težav z uporabo

algoritmov za enakomerno porazdeljevanje podatkov po gruči in pomnilniških naprav, ki so zmožne samodejno uravnotežiti porazdelitev shranjenih podatkov po vseh napravah v gruči. Enakomerna porazdelitev je pomembna za zagotovitev optimalnega delovanja sistema pod večjo obremenitvijo. Posebnost RADOSa je v navidezno naključni (angl. *pseudo random*) distribuciji predmetov po pomnilniku. Postavitev in vpogled predmetov poteka algoritmično, in se ob vsaki transakciji znova izračuna.

Shramba RADOS temelji na kazalu gruče (angl. *cluster map*), ki določa, katere naprave OSD so prisotne v gruči in opisuje razporeditev podatkov med temi napravami. Kazalo vzdržuje gruča monitorjev, ki skrbijo za dostopnost naprav OSD in v primeru spremembe stanja to v kazalu zabeležijo. Kazalo gruče poznajo vsi strežniki in odjemalci gruče Ceph. Ob vsaki spremembi kazala se poveča generacija kazala (angl. *epoch*). S to vrednostjo odjemalci vzdržujejo lokalno shranjeno stanje gruče in razporeditev podatkov. Vsaka sprememba kazala se porazdeli po celotni gruči [12]. Kazalo gruče je podrobneje opisano v poglavju 4.3.

RADOS nudi različne vmesnike shrambe z uporabo knjižnice `librados`. To knjižnico v Cephu uporabljata prehod RADOS, ki nudi predmetni vmesnik, združljiv z S3 in RADOS bločna naprava (angl. *block device*) RBD, ki jo kot bločno napravo priključimo odjemalcu. Knjižnico librados lahko uporabimo za razvoj lastne aplikacije, ki podatke shranjuje kot predmete. Za shranjevanje podatkov lahko uporabimo še datotečni sistem Ceph FS, ki je skladen z vmesnikom POSIX. Slika 4.2 prikazuje vmesnike shrambe RADOS in odvisnosti med njimi. Datotečni sistem Ceph FS komunicira neposredno s shrambo RADOS. Aplikacijam je omogočen dostop z uporabo knjižnice librados. Na tej knjižnici temeljita vmesnika prehod RADOS (*RADOSGW*) in bločna naprava RADOS (*RBD*).

Podatki so v RADOSu shranjeni v bazene (angl. *pool*), logične skupine, ki omogočajo bolj natančen nadzor nad celotno gručo, saj lahko na nivoju bazena nastavljamo željeno stopnjo replikacije in podatke ločujemo med seboj.

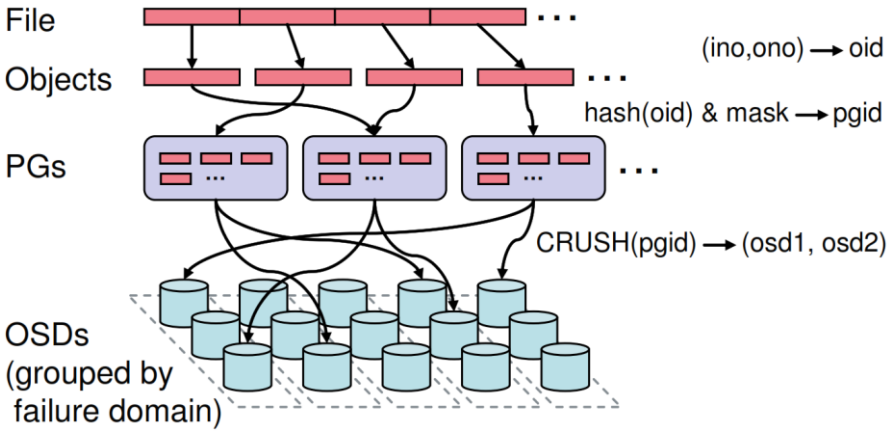


Slika 4.2: Vmesniki do predmetne shrambe RADOS [13].

4.1.1 Postavitev podatkov

Predmeti v shrambi RADOS so naslovljivi podobno kot pari ključev in vrednosti v slovarju, vsak ključ enolično določa predmet. Posebnost RADOSa je, da se lokacija ključa ob vsakem vpogledu in shranjevanju izračuna, s čimer se izognemo vzdrževanju centralizirane tabele dodeljevanja. Ta algoritem je poznan vsakemu članu gruče Ceph. Vsaka datoteka, ki jo shranimo, se najprej razdeli na več predmetov vnaprej določene velikosti.

Slika 4.3 prikazuje potek shranjevanja na pomnilniške naprave. Vsaka datoteka (angl. *file*) se razdeli na predmete, katerim se določi oznaka *oid*. Zgoščevalna funkcija *hash*, umesti *oid* v postavitveno skupino *pgid*. Zgoščevalna funkcija se izračuna na podlagi imena predmeta, željene stopnje replikacije in števila vseh postavitvenih skupin. Algoritem CRUSH za vsako postavitveno skupino *pgid* izračuna seznam predmetnih shranjevalnih naprav *osd*, kjer bodo replike postavitvenih skupin shranjene.



Slika 4.3: Postopek postavitve predmeta na ustrezne naprave OSD [14].

4.1.2 Algoritem CRUSH

CRUSH (Controlled Replication Under Scalable Hashing) je navidezno naključen algoritem za porazdelitev predmetov po strukturirani pomnilniški gruči [15]. Definiran je kot deterministična funkcija, kar pomeni, da za iste vhodne parametre vedno vrne enak rezultat. Funkcija za vhodni podatek, ki je v primeru RADOSa ID postavitvene skupine (na sliki Slika 4.3 oznaka *pgid*), izračuna seznam naprav OSD (na sliki označen z *(osd1, osd2)*), kjer se replike shranijo. Pri izračunu upošteva uteži in hierarhijo naprav, ki je definirana v kazalu CRUSH (Izpis 4.1) in stremi k temu, da so replike porazdeljene ne le med različnimi OSDji temveč predvsem med različnimi strežniki, s čimer se v primeru odpovedi strojne opreme izognemo izgubi podatkov. Ker je funkcija deterministična, jo lahko poleg shranjevanje uporabimo tudi za iskanje predmetov v gruči. Za izračun lokacije katerega koli predmeta mora algoritem poznati le podatke o postavitveni skupini in tabelo OSDjev.

```
# buckets
host ceph2 {
        id -2           # do not change unnecessarily
        # weight 1.300
        alg straw
        hash 0  # rjenkins1
        item osd.0 weight 0.450
        item osd.6 weight 0.850
}
root default {
        id -1           # do not change unnecessarily
        # weight 5.330
        alg straw
        hash 0  # rjenkins1
        item ceph2 weight 1.300
        item ceph3 weight 1.310
        item ceph4 weight 1.820
        item ceph1 weight 0.900
}
```

Izpis 4.1: Uteži OSDjev strežnika ceph2 in skupne uteži vseh strežnikov v gruči, zabeležene v kazalu CRUSH.

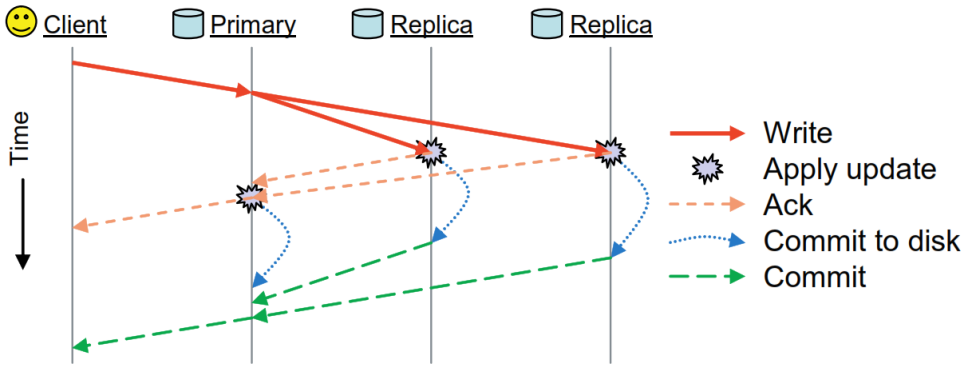
Ker se lokacija vsakega predmeta sproti izračuna s postavitvijo v skupino in algoritmom CRUSH, je količina meta podatkov, potrebnih za vzdrževanje takega sistema, zelo majhna. Meta podatki se spreminjajo le ob dodajanju in odstranjevanju pomnilniških naprav, ker je potrebno novo dodane naprave obtežiti in na novo definirati kazalo CRUSH. Ko dodamo novo napravo, algoritem izbere naključen izbor shranjenih predmetov, in jih dodeli v nove postavitvene skupine in shrani na novo dodano napravo.

4.2 Naprava za shranjevanje predmetov Ceph-OSD

Shrambo RADOS sestavljajo pomnilniške naprave OSD ki predmete shranjujejo na bločne naprave. Za vsako napravo OSD skrbi aplikacija Ceph-OSD, s katero se odjemalci neposredno povežejo za operacije branja in pisanja. Ker so povezave neposredne, brez posredniških strežnikov, je princip delovanja Cepha podoben delovanju omrežja vsak z vsakim (angl. *peer to peer*) ali krajše P2P. Tak sistem odpravi možnost ozkega grla s posredniškimi strežniki, preko katerih bi se odjemalci povezovali. Možnost neposredne povezave omogoča ravno celovito, redno osveženo kazalo gruče in algoritem CRUSH, o katerem smo govorili v poglavju 4.1.1. Odjemalec si mora pred začetkom operacije od gruče monitorjev zagotoviti zadnjo generacijo kazala, s katerim izračuna lokacijo predmeta, ki ga želi prebrati ali novo lokacijo, kamor naj predmet zapiše. Rezultat je naslov naprave, ali več naprav Ceph-OSD, na katere se odjemalec

priključi in izvede vhodno izhodne operacije [14]. Ceph strogo ločuje med podatki in meta podatki, kar prikazuje že omenjena Slika 4.1. Meta podatkov, ki so potrebni za vzdrževanja imenskega prostora in varnosti v datotečnem sistemu CephFS, ne shranjujejo OSDji, ampak jih obravnava strežnik za meta podatke MDS.

Ceph-OSD sodeluje pri replikaciji predmetov. Ko CRUSH določi seznam OSDjev, kamor se predmet shrani, vedno določi primarni OSD. Odjemalec se poveže le s primarnim OSDjem, kamor predmet zapiše. Naloga primarnega OSDja je, da ta predmet zapiše še na vse replike. Slika 4.4 opisuje postopek shranjevanja predmeta na glavni OSD in vse replike. Šele ko predmet uspešno doseže vse replike, se predmet dejansko shrani na disk in odjemalec prejme potrdilo o uspešni transakciji. Slabost takega pristopa shranjevanja je zakasnitev, ki se pojavi zaradi distribucije predmeta po vseh replikah, ki je v hitrih omrežjih in zmogljivi strojni opremi zanemarljiva, prednost pa je zagotovitev odjemalcu, da je predmet res shranjen v več replikah. Ceph omogoča nastavitev replikacije za vsak bazen posebej.

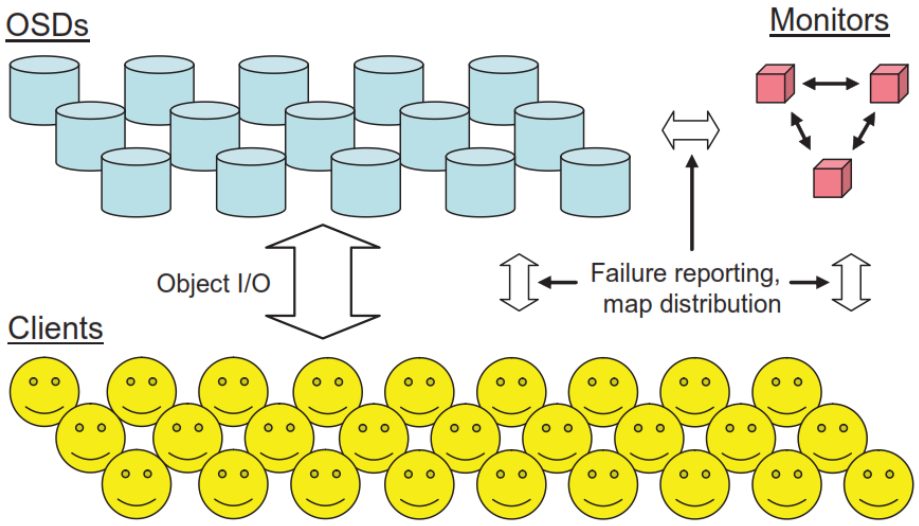


Slika 4.4: Replikacija predmetov v shrambi RADOS.

4.3 Monitor Ceph-MON

Monitorji vzdržujejo glavno kopijo kazala gruče Ceph. Kazalo gruče je sestavljena iz seznama monitorjev, seznama OSDjev, seznama postavitvenih skupin in seznama strežnikov za meta podatke. Vsak odjemalec lahko s povezavo na enega izmed monitorjev pridobi zadnji različico kazala gruče, iz katerega razbere lokacijo vsakega strežnika v gruči. Z zadnjo generacijo kazala in algoritmom CRUSH lahko vsak odjemalec izračuna lokacijo katerega koli predmeta, shranjenega v gruči. Kazalo gruče beleži, kateri procesi so prisotni (angl. *in*) in kateri izmed prisotnih so dosegljivi (angl. *up*). Monitorji v kazalu postavitvenih skupin beležijo vsako spremembo stanja, v primeru da postavitvena skupina postane neaktivna ali če je v stanju obnovitve zaradi nesreče. Poleg vzdrževanja kazala, monitorji skrbijo še za overjanje dostopa in beleženje dnevnikov dogodkov [16]. Slika 4.5 prikazuje položaj in vlogo monitorjev v gruči.

Monitorji med odjemalci (angl. *clients*) in OSDji posredujejo obvestila o napakah (angl. *failure reporting*) in kazalo gruče (na sliki angl. *map distribution*).



Slika 4.5: Monitorji v sistemu Ceph skrbijo za nadzor sistema, poročanje o nesrečah in distribucijo kazala elementov celotne gruče [12].

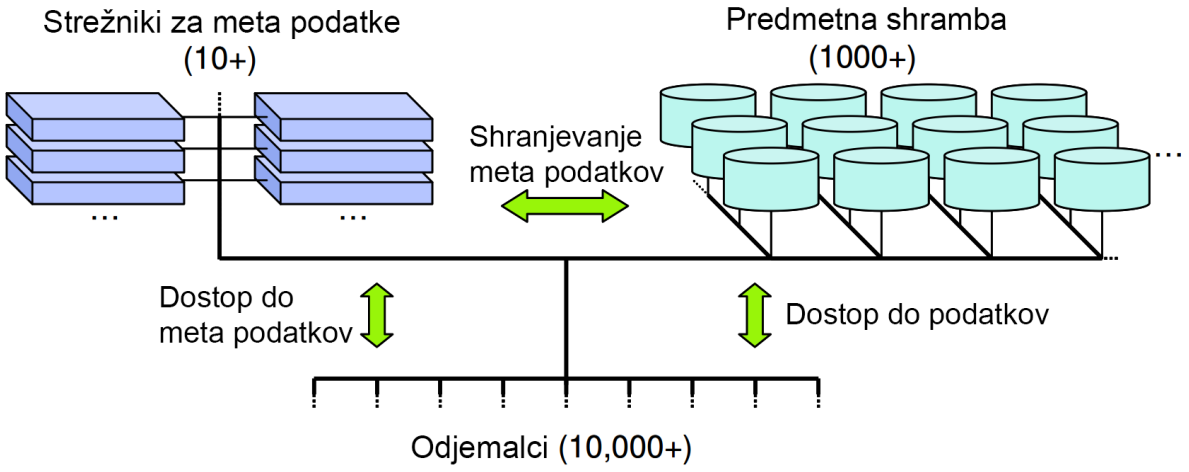
4.3.1 Protokol Paxos

Za visoko razpoložljivost Cepha potrebujemo prisotno gručo vsaj treh monitorjev, ker je en sam monitor možna točka odpovedi. Kadar je v gruči prisotnih več monitorjev, ti za dosego soglasja o trenutnem kazalu gruče uporabljajo protokol Paxos, ki je poenostavljen tako, da dovoljuje le eno spremembo kazala hkrati. Gruča monitorjev izvoli voditelja, ki vodi posodobitve kazala in upravlja skladnost. Ko je voditelj izbran, od vsakega monitorja v gruči pridobi shranjeno generacijo kazala. Vsak monitor ima na voljo T enot časa (v trenutni različici Cepha dve sekundi), da se na poziv odzove. Skupina monitorjev je sklepčna, če se pravočasno odzove vsaj večina. Če je aktivna večina monitorjev, Paxos poskrbi, da ima vsak izmed monitorjev shranjeno najnovejšo različico kazala gruče, sicer si od ostalih monitorjev pridobi delne popravke. Voditelj vsem aktivnim monitorjem za kratek čas dovoli distribucijo kopije kazala OSDjem in odjemalcem, ki kazalo zahtevajo. Če se po T časa dovoljenje ne obnovi, se sklepa, da je voditelj nedosegljiv in monitorji izvolijo novega voditelja. Ko aktiven monitor prejme kazalo, najprej preveri, da je generacija kazala res novejša, kot njegovo shranjeno kazalo. Če je kazalo novejše, svojega shranjenega dopolni s spremembami. Generacija kazala se z vsako spremembo veča, nikoli ne manjša, kar zagotovi da monitor nikoli ne popravi kazala na starejšo generacijo v primeru zamude v omrežju [14] [17].

4.4 Datotečni sistem CephFS in strežnik za meta podatke

Ceph poleg predmetne shrambe odjemalcem nudi datotečni sistem CephFS, ki je skladen s POSIX. Na poizvedbe po meta podatkih odgovarja posebna gruča strežnikov Ceph-MDS, ki v datotečnem sistemu vzdržujejo strukturo imenikov in pravice nad datotekami. Pred vsako vhodno izhodno operacijo se za pravice dostopa do datoteke odjemalec poveže s strežnikom Ceph-MDS. Če odjemalec dobi dovoljenje za dostop, se naknadne operacije branja in pisanja opravijo neposredno z OSDji. Ločevanje podatkov in meta podatkov ima velik vpliv na zmogljivost celotnega sistema, saj v datotečnem sistemu operacije nad meta podatki predstavljajo več kot polovico vseh transakcij, čeprav količinsko zasedajo le majhen del podatkov. Slika 4.6 prikazuje arhitekturo dostopa do meta podatkov in do predmetne shrambe. Odjemalci meta operacije opravljajo s strežniki za meta podatke, za shranjevanje podatkov pa se povezujejo neposredno z gručo predmetne shrambe. Strežniki za meta podatke imajo vlogo predpomnilnika za shranjevanje meta podatkov datotečnega sistema, sčasoma pa vse meta podatke kot predmete shranijo na predmetne shranjevalne naprave [18] [19].

Odjemalci, ki uporabljajo datotečni sistem CephFS, se v gručo Ceph povežejo neposredno z OSDji. CephFS lahko na strežnik priklopimo z gonilnikom v jedru ali kot datotečni sistem v uporabniškem načinu FUSE.

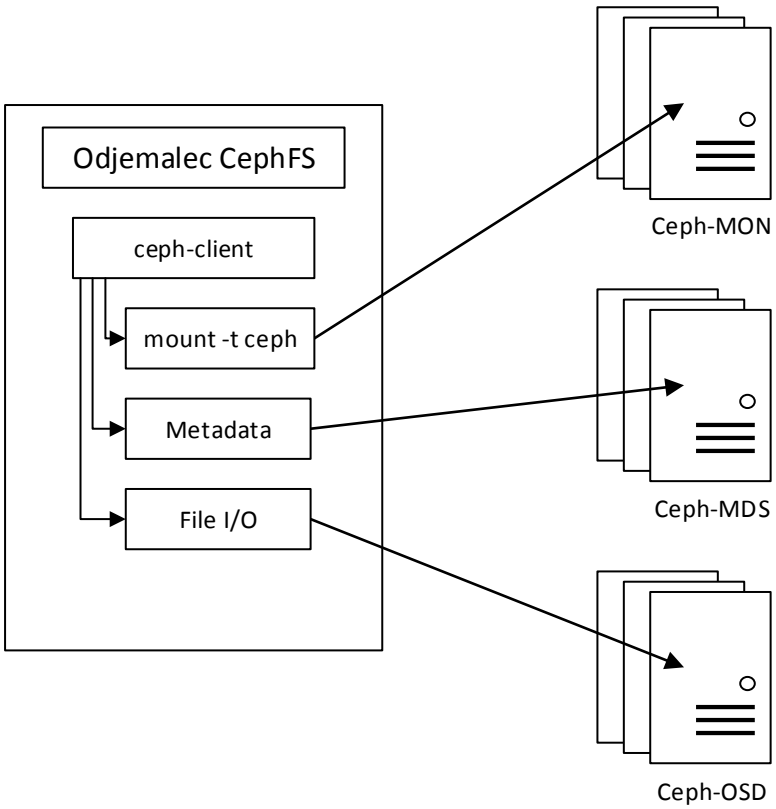


Slika 4.6: Arhitektura dostopa do meta podatkov in do shrambe podatkov v porazdeljenem datotečnem sistemu CephFS [18].

4.4.1 Priključitev datotečnega sistema na odjemalca

Datotečni sistem CephFS lahko na odjemalca priključimo preko vmesnika `ceph-client` v jedru operacijskega sistema ali kot datotečni sistem v uporabniškem načinu, ki smo ga že

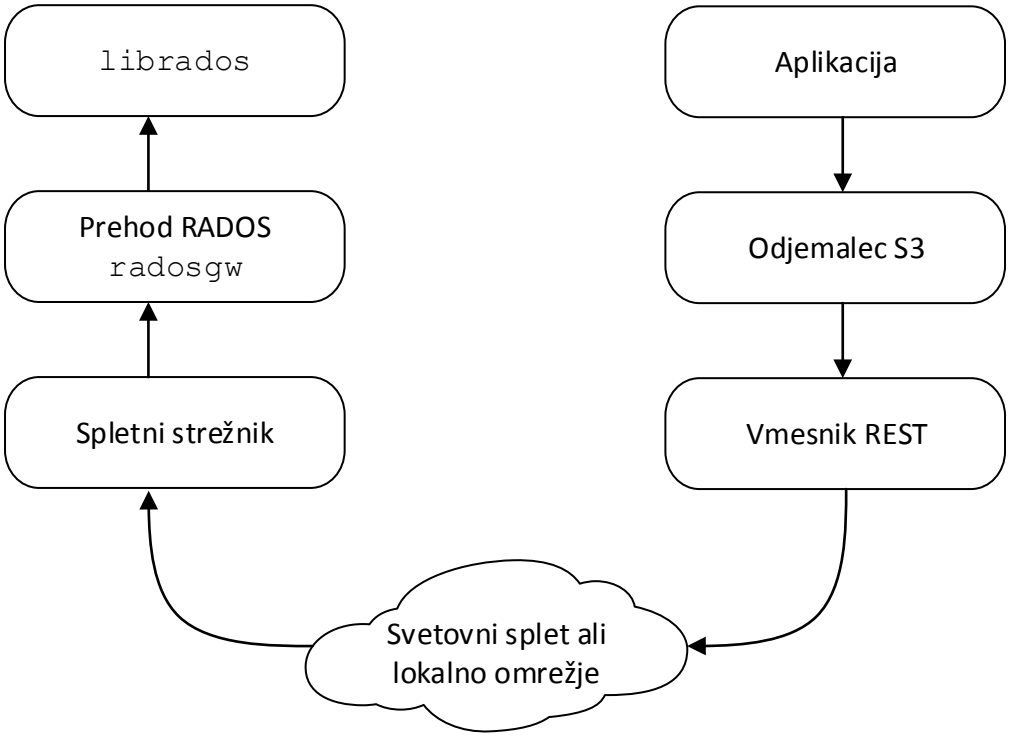
omenili v poglavju 3.3. Slika 4.7 prikazuje priključitev in tok podatkov med odjemalcem in datotečnim sistemom CephFS. Odjemalec datotečnega sistema CephFS ceph-client se za priključitev shrambe s ključem overi pri monitorjih Ceph-MON, za poizvedbe meta podatkov komunicira s strežniki Ceph-MDS, vhodno izhodne operacije pa izvaja neposredno z gručo Ceph-OSD. Namestitev datotečnega sistema CephFS in priključitev na odjemalca je opisana v dodatku C.



Slika 4.7: Komunikacija odjemalca CephFS s strežniki gruče Ceph.

4.5 Prehod RADOS

Prehod RADOS (angl. *RADOS Gateway*) je storitev, ki odjemalcem nudi dostop do predmetne shrambe preko programskega vmesnika REST, ki je združljiv z S3. Slika 4.2 prikazuje položaj prehoda RADOS (na sliki *RADOSGW*) v gruči Ceph. RADOSGW s shrambo RADOS govori preko knjižnice `librados`. RADOSGW sam vzdržuje podatke o uporabnikih prehoda, skrbi za overjanje uporabnikov in nadzoruje dostop do podatkov. Slika 4.8 prikazuje način povezave odjemalca s prehodom RADOS. Aplikacija preko programskega vmesnika REST na spletni strežnik pošlje poizvedbe v obliki, primerni za shrambo S3. Spletni strežnik poizvedbe posreduje prehodu RADOS, ki podatke preko programske knjižnice `librados` shrani v predmetno shrambo RADOS.



Slika 4.8: Dostop do predmetne shrambe preko prehoda RADOS z uporabo programskega vmesnika REST.

Poglavje 5 Primerjave in meritve

V tem poglavju bomo prikazali več primerjav med datotečnimi sistemi in protokoli, ki smo jih opisali v prejšnjih poglavjih. Primerjali bomo hitrost in čas prenosa ter porabljen procesorski čas v uporabniškem in jedrnem načinu. Prenašali in brisali bomo eno veliko (1,3 GB) in več manjših datotek (1190 datotek, 246 MB).

Testno okolje sta sestavljala strežnik in odjemalec, povezana preko gigabitne omrežne povezave. Izmerjena hitrost povezave med računalnikoma je bila 896 Mbit/s. Specifikacije računalnikov opisuje Tabela 5.1. Do spletne shrambe Amazon S3 smo z odjemalca dostopali preko spletne povezave z izmerjeno hitrostjo sprejemanja 537 Mbit/s in s hitrostjo oddajanja 260 Mbit/s.

Lastnosti	**Strežnik**	**Odjemalec**
Operacijski sistem	Linux, Ubuntu Server 14.04	Linux, Ubuntu 15.04
Centralna procesorska enota	intel Celeron N3150 @ 2,08GHz	intel Core i7-4790 @ 4GHz
Delovni pomnilnik	16 GB	16 GB
Hitrost trdega diska	319 MB/s	1,1 GB/s

Tabela 5.1: Specifikacije računalnikov, uporabljenih v testnem okolju.

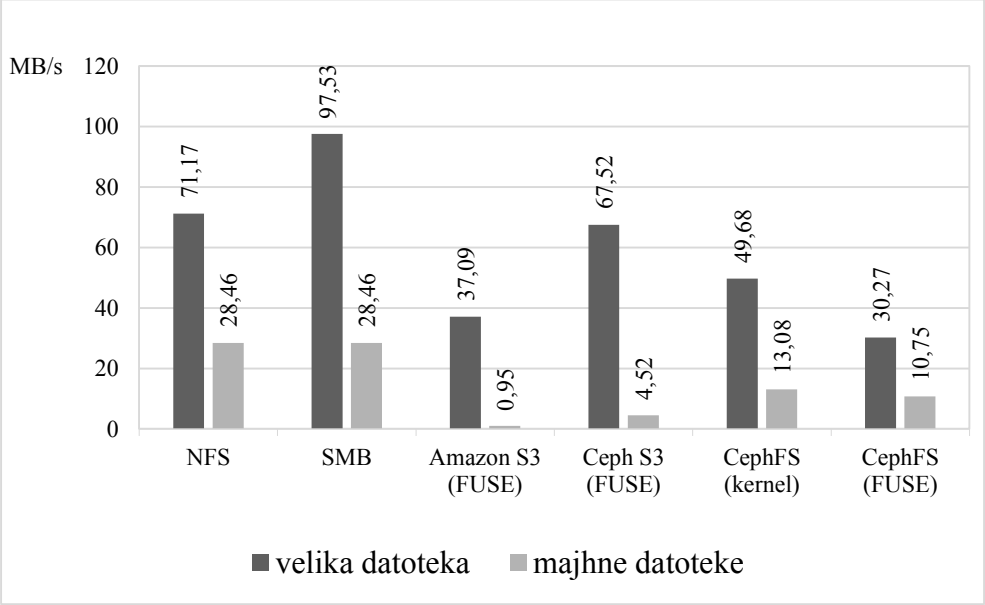
Za merjenje časa smo uporabili ukaz `time`, ki meri čas izvajanja, uporabniški čas, ki ga procesorska enota porabi za izvajanje procesa v uporabniškem načinu in sistemski čas, ki predstavlja čas, ki ga procesor porabi za izvajanje kode v jedru, na primer pri sistemskih klicih procesa. Za prenos datotek in merjenje hitrosti smo uporabili ukaz `rsync -av`, ki rekurzivno prenaša datoteke in izpisuje stanje prenosa ter na koncu izpiše povprečno hitrost prenašanja. Datoteke smo brisali z ukazom `rm -r`. Vsak tip shrambe smo na računalnik priključili z ukazom `mount` v jedrnem načinu, ali z drugim ustreznim ukazom v primeru priključitve datotečnega sistema v uporabniškem načinu (FUSE). Do shrambe S3 z vmesnikom REST smo dostopali s programom `s3cmd` [20]. Predmete smo prenašali z ukazoma `put` in `get`, brisali smo jih z ukazom `delete`. Hitrost prenosa smo merili v megabajtih na sekundo [MB/s], čas prenosa in brisanja pa v sekundah [s].

5.1 Primerjava datotečnih sistemov

Opravili smo primerjave med datotečnimi sistemi NFS, SMB in CephFS, ki smo jih priključili preko gonilnikov v jedru operacijskega sistema in med datotečnimi sistemu, priključenimi v uporabniškem načinu (FUSE) s3fs (Amazon S3), s3fs (Ceph S3) in CephFS.

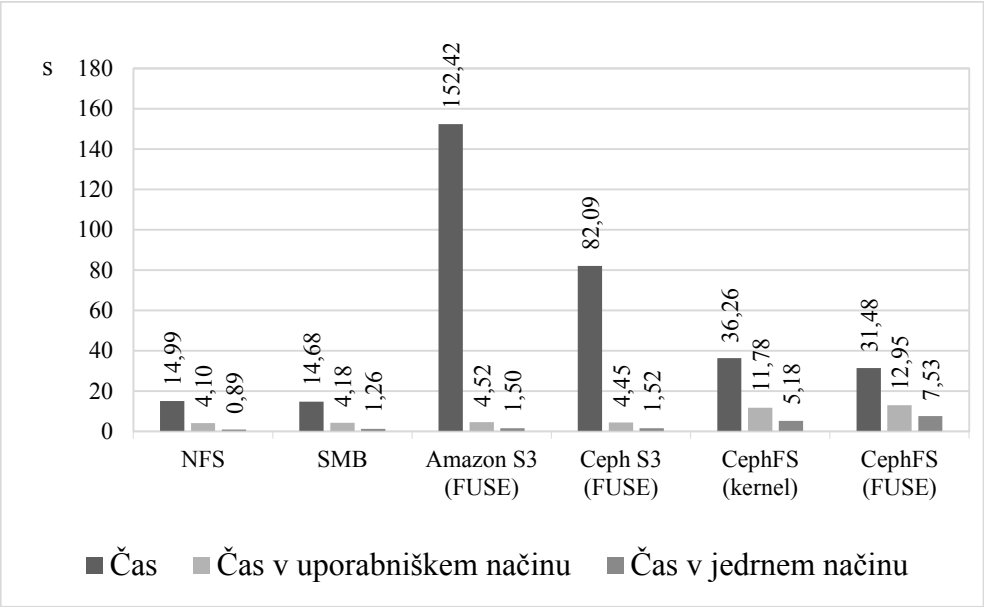
5.1.1 Zapisovanje podatkov v datotečnih sistemih

Slika 5.1 prikazuje hitrosti zapisovanja velike in majhnih datotek v megabajtih na sekundo. Pri zapisovanju sta se najbolje odrezala protokola NFS in SMB, kar je bilo pričakovano, saj sta oba protokola hitra in enostavna. Pri prenosu datoteke v shrambo Ceph preko katerega koli vmesnika, se vsi podatki shranijo v treh izvodih. Ker Ceph odjemalcu vrne potrditev zapisa predmeta v shrambo šele, ko so vse kopije predmeta shranjene, je hitrost prenosa temu ustrezno manjša. Shramba s3fs je dosegla slabše rezultate predvsem pri prenosu veliko manjših datotek zaradi prenašanja preko protokola HTTP. Za vsako datoteko, ki jo shranimo kot predmet, se mora poslati nova zahteva, kar pri veliki količini majhnih datotek pomeni zelo veliko meta podatkov in s tem slabšo hitrost prenosa. Pri prenosu ene same velike datoteke se je datotečni sistem s3fs izkazal zadovoljivo, v primeru Cepha celo hitreje kot CephFS. Če primerjamo še različna načina priključitve CephFS, opazimo da gonilnik FUSE pomeni padec v zmogljivosti zaradi dodatnega vmesnika, kot je opisan v poglavju 3.3.

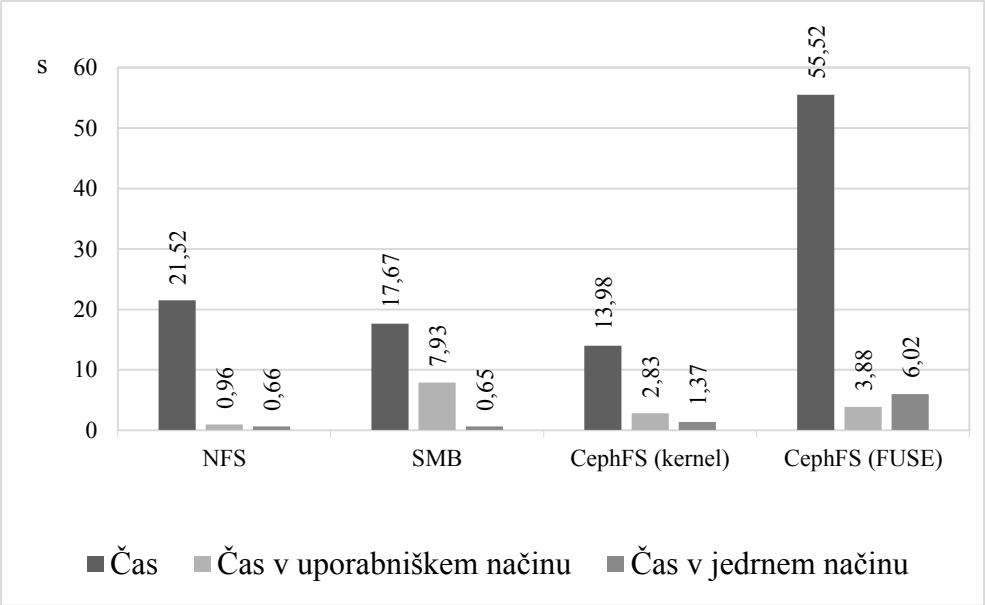


Slika 5.1: Hitrost zapisovanja datoteke v datotečni sistem.

Slika 5.2 prikazuje čas sekundah, potreben za prenos velike datoteke v datotečni sistem. Prenos je bil najhitrejši preko protokolov NFS in SMB. Prenos preko s3fs je trajal najdlje. Kljub temu, da so bile dosežene hitrosti prenosa visoke, se je veliko časa porabilo zaradi protokola HTTP, čas procesiranja pa je bil visok zaradi prevajanja POSIX ukazov v ukaze REST, kar počne s3fs. CephFS je v obeh primerih priključitve dosegel podoben rezultat, ki je bil višji kot pri NFS in SMB. Čas procesiranja je bil visok, ker mora odjemalec datotečnega sistema Ceph sproti računati, kam predmet shraniti.

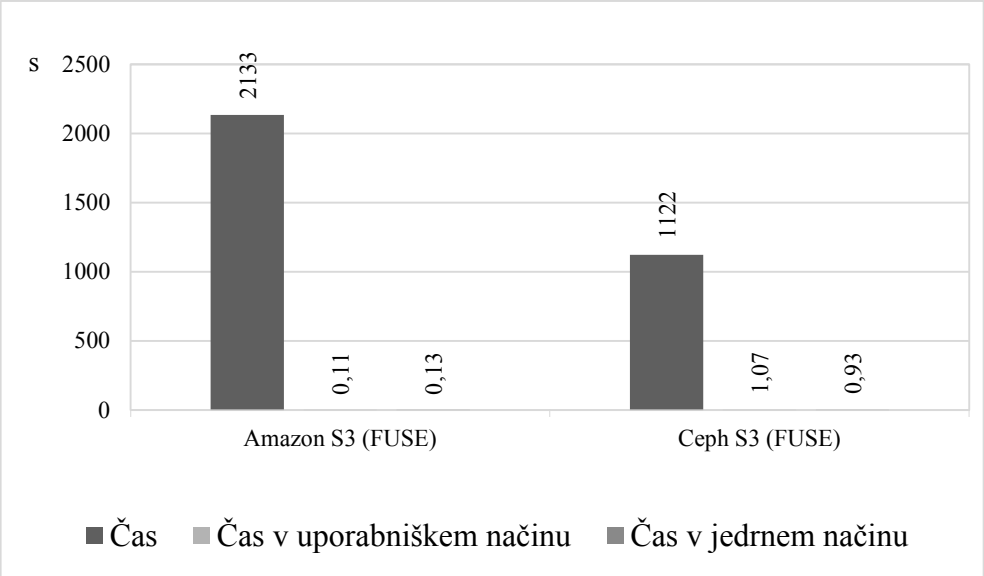


Slika 5.2: Čas zapisovanja velike datoteke v datotečni sistem.



Slika 5.3: Čas zapisovanja majhnih datotek v datotečni sistem.

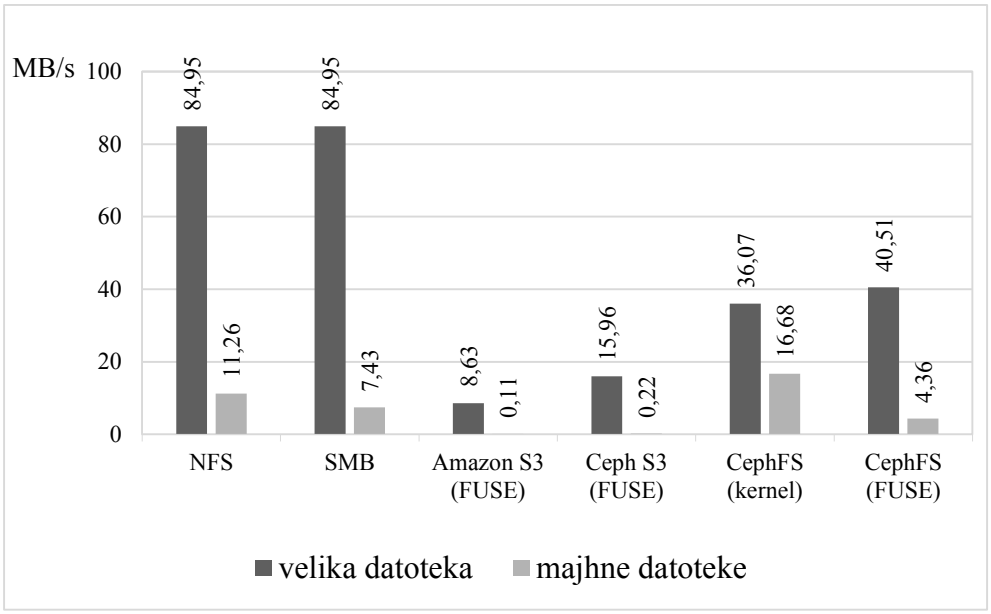
Slika 5.3 prikazuje rezultate NFS, SMB in CephFS. V tem primeru se je CephFS, priključen preko gonilnika v jedru izkazal najbolje, kljub bolj zahtevnemu procesiranju kot pri protokolu NFS. Zapisovanje v CephFS, priključen kot FUSE je trajalo najdlje. Rezultat zapisovanja v datotečni sistem s3fs smo ločeno prikazali na **Error! Reference source not found.**. Prenos veliko manjših datotek v shrambo Amazon S3, priključeno z s3fs je trajal kar 35 minut, pri shrambi Ceph S3 z s3fs pa več kot 18 minut.



Slika 5.4: Čas zapisovanja majhnih datotek v datotečni sistem s3fs.

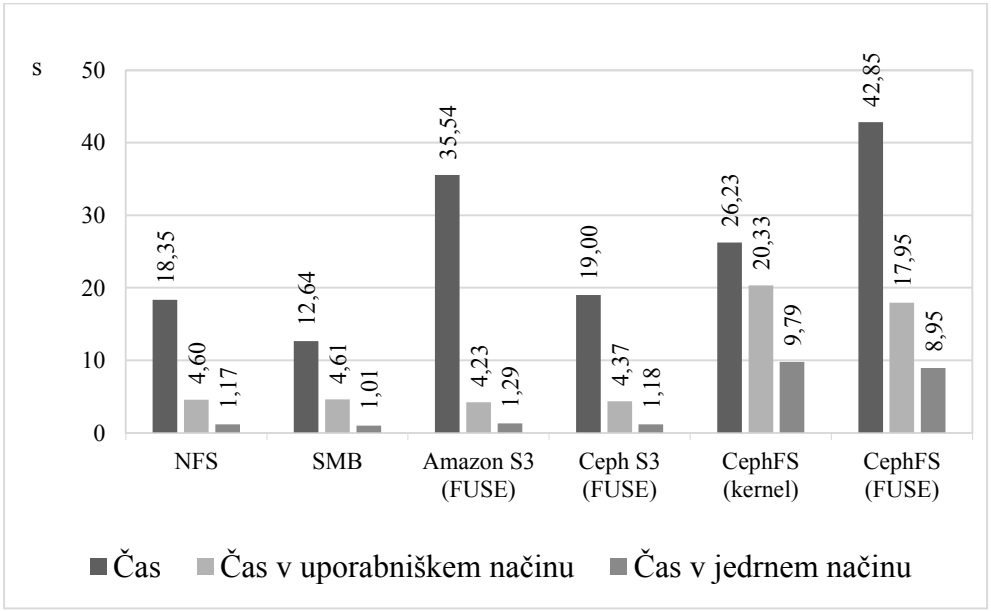
5.1.2 Branje podatkov iz datotečnega sistema

Slika 5.5 prikazuje hitrosti branja velike in majhnih datotek v megabajtih na sekundo. Kot pri zapisovanju, sta se pri branju velike datoteke najbolje odrezala protokola NFS in SMB. Prenos majhnih datotek je bil najhitrejši pri datotečnem sistemu CephFS. Ker Ceph predmete shranjuje porazdeljeno med več pomnilniškimi napravami, lahko vsak predmet prebere iz tiste naprave, ki je trenutno najmanj obremenjena. Ker so predmeti shranjeni v več kopijah, lahko Ceph predmet prebere iz katerekoli od naprav, ki kopijo iskanega predmeta vsebuje. Na ta način Ceph dobro izkoristi strojno opremo, ki je na voljo, za doseganje boljše zmogljivosti sistema. Shramba s3fs se je v pri branju izkazala za najpočasnejšo, predvsem pri branju majhnih datotek.

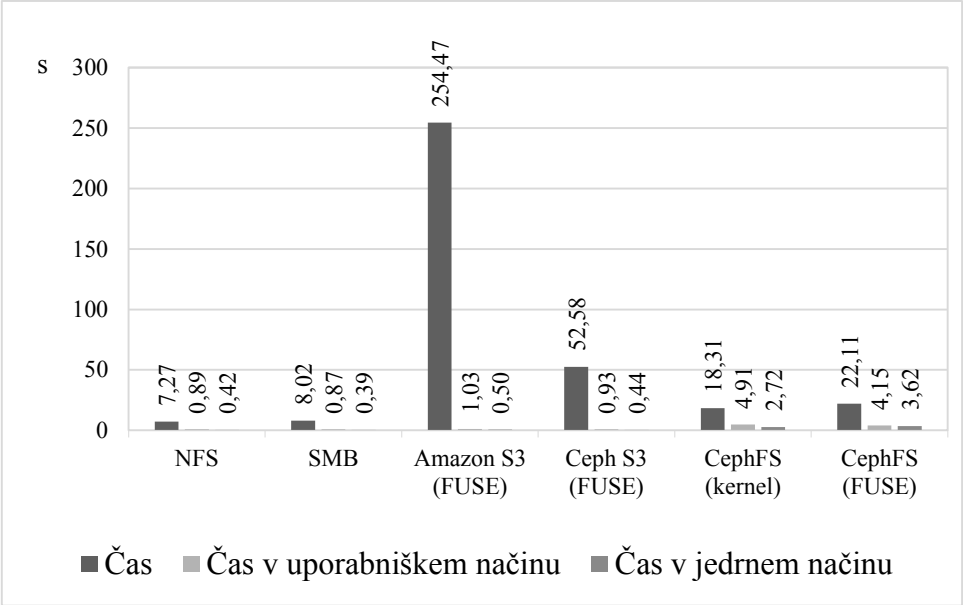


Slika 5.5: Hitrost branja datoteke iz datotečnega sistema.

Iz slik Slika 5.6 in Slika 5.7 razberemo čas in čas procesiranja, ki je bil potreben za prenos podatkov. Protokola NFS in SMB za procesor nista zelo zahtevna, glede na hitrost prenosa, ki jo ponujata. Opazimo lahko, da je CephFS za odjemalca zahteven za procesiranje, saj mora lokacije OSDjev, kjer so predmeti shranjeni, izračunati odjemalec sam.



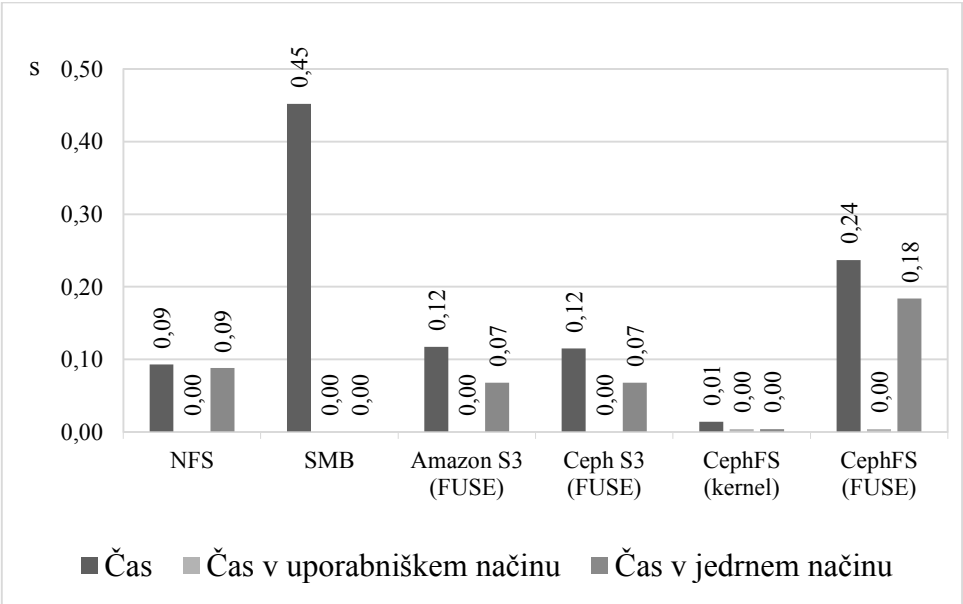
Slika 5.6: Čas branja velike datoteke iz datotečnega sistema.



Slika 5.7: Čas branja majhnih datotek iz datotečnega sistema.

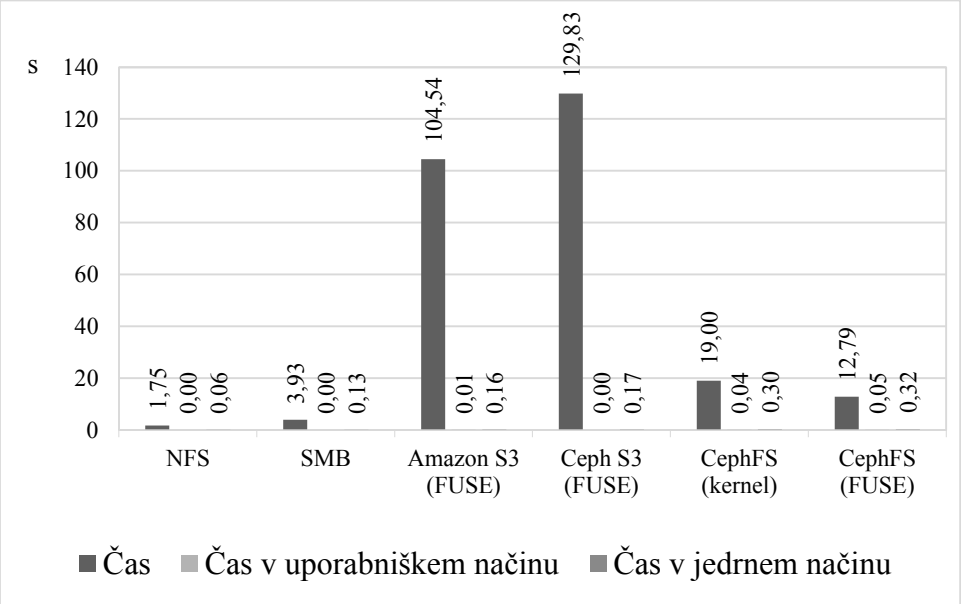
5.1.3 Brisanje podatkov iz datotečnega sistema

Primerjali smo brisanje datotek iz datotečnih sistemov. Slika 5.8 prikazuje čase brisanja velike datoteke v sekundah glede na shrambo. Čas izbrisa je pri vseh shrambah zanemarljiv, in je v najslabšem primeru, pri protokolu SMB, manj kot pol sekunde.



Slika 5.8: Čas brisanja velike datoteke iz datotečnega sistema.

Slika 5.9 prikazuje čase brisanja majhnih datotek v sekundah. Pri datotečnem sistemu CephFS je operacija trajala od 13 do 19 sekund. Zanimivo je, da je boljši rezultat dosegel CephFS, priključen preko FUSE. Izbris podatkov iz s3fs je trajal najdlje.



Slika 5.9: Čas brisanja majhnih datotek iz datotečnega sistema.

5.2 Primerjava prenosa do shrambe S3 preko vmesnika REST

S programom `s3cmd` smo merili hitrosti prenosa podatkov v shrambo Amazon S3 in Ceph S3 preko vmesnika REST, brez priključitve kot datotečni sistem. Prenašali smo iste datoteke kot v poglavju 5.1, in sicer eno veliko datoteko velikosti 1,3 GB in več manjših datotek skupne velikosti 246 MB (1190 datotek). Vse meritve smo opravili iz računalnika odjemalec, opisanega v začetku poglavja Poglavje 5. Za prenašanje podatkov s strežnika smo uporabili funkcijo `get`, za nalaganje podatkov na strežnik funkcijo `put`, za brisanje pa funkcijo `delete`. Izpis 5.1 prikazuje uporabo ukaza `s3cmd` z funkcijo `put`. Datoteka se razdeli na dele, velike 15MB, ki se prenesejo v predmetno shrambo.

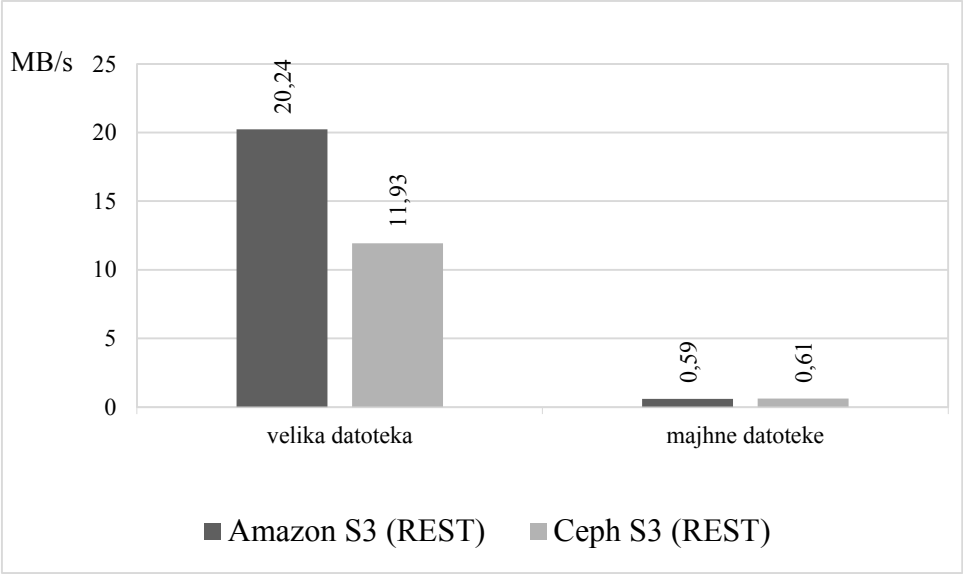
```
jan@itpc:~/mounting_scripts$ s3cmd -c .cephs3cfg put kubuntu.iso s3://TEST
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 1 of 88, 15MB]
 15728640 of 15728640   100% in    0s    22.97 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 2 of 88, 15MB]
 15728640 of 15728640   100% in    0s    22.27 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 3 of 88, 15MB]
 15728640 of 15728640   100% in    0s    18.18 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 4 of 88, 15MB]
 15728640 of 15728640   100% in    0s    18.65 MB/s  done
 ...
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 83 of 88, 15MB]
 15728640 of 15728640   100% in    0s    20.36 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 84 of 88, 15MB]
 15728640 of 15728640   100% in    0s    16.77 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 85 of 88, 15MB]
 15728640 of 15728640   100% in    1s    13.05 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 86 of 88, 15MB]
 15728640 of 15728640   100% in    1s     8.04 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 87 of 88, 15MB]
 15728640 of 15728640   100% in    0s    16.03 MB/s  done
kubuntu.iso -> s3://TEST/kubuntu.iso  [part 88 of 88, 11MB]
 11878400 of 11878400   100% in    0s    21.69 MB/s  done
jan@itpc:~/mounting_scripts$
```

Izpis 5.1: Ukaz s3cmd put za prenos datoteke na strežnik S3 in delni izpis.

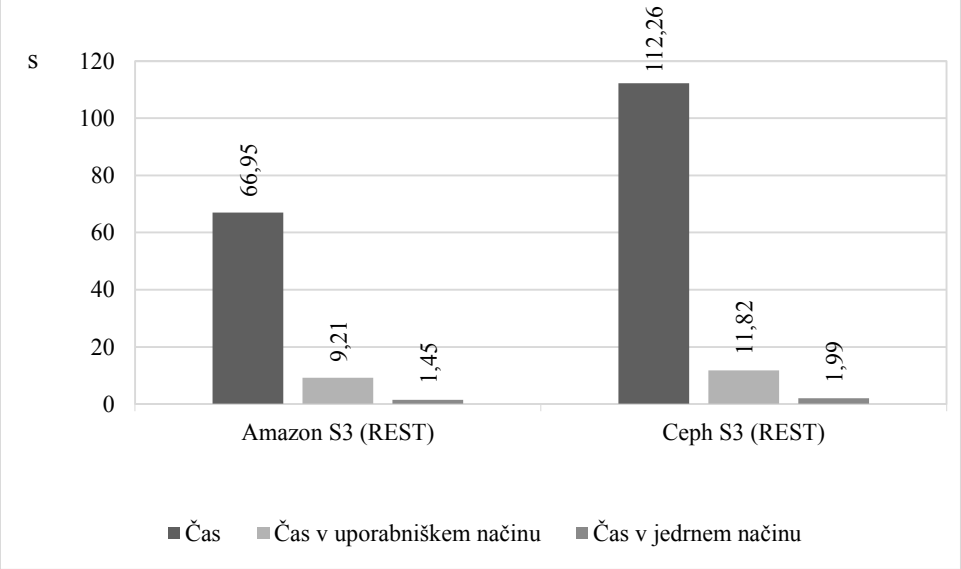
5.2.1 Zapisovanje predmeta v shrambo S3

Slika 5.10 prikazuje hitrost prenosa velike in male datoteke v shrambo Amazon S3 in Ceph S3 preko vmesnika REST. Enota hitrosti je megabajt na sekundo.

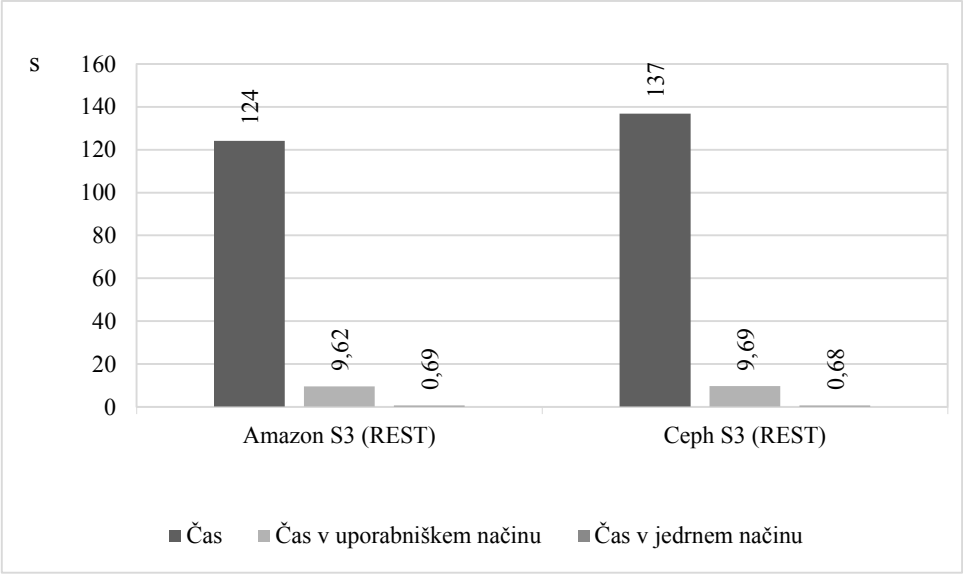
Sliki Slika 5.11 in Slika 5.12 prikazujeta čas prenosa datotek na strežnik. Pri prenosu velike datoteke je bila hitrejša shramba Amazon S3. Pri prenosu majhnih datotek sta bili obe shrambi približno enako počasni.



Slika 5.10: Hitrost zapisovanja (PUT) predmeta preko vmesnika REST.



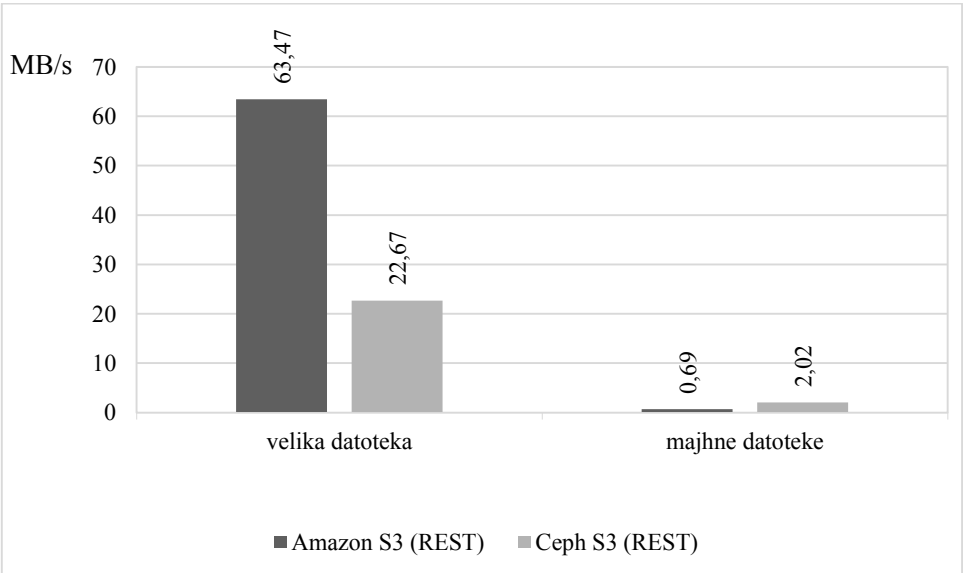
Slika 5.11: Čas zapisovanja (PUT) velikega predmeta preko vmesnika REST.



Slika 5.12: Čas zapisovanja (PUT) majhnih predmetov preko vmesnika REST.

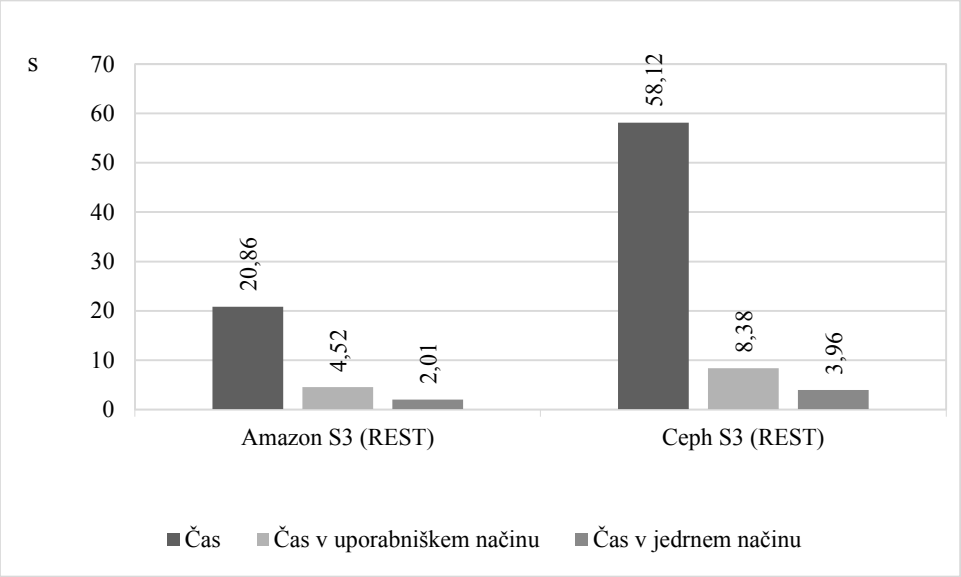
5.2.2 Branje predmeta iz shrambe S3

Slika 5.13 prikazuje hitrost prenosa datotek v smeri odjemalca. Hitrost prenosa velike datoteke je bila višja pri shrambi Amazon S3, prenos majhnih datotek pa je bolje obdelal Ceph S3.

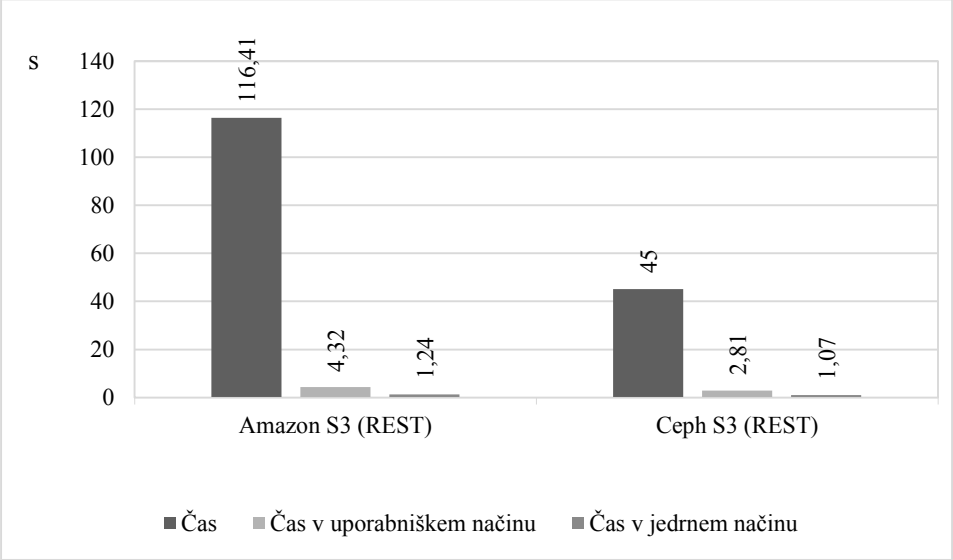


Slika 5.13: Hitrost branja (GET) predmeta preko vmesnika REST.

Sliki Slika 5.14 in Slika 5.15 prikazujeta čas prenosa datotek v smeri odjemalca. Prenos velike datoteke je bil hitreje opravljen iz shrambe Amazon S3, prenos majhnih datotek pa iz shrambe Ceph S3.



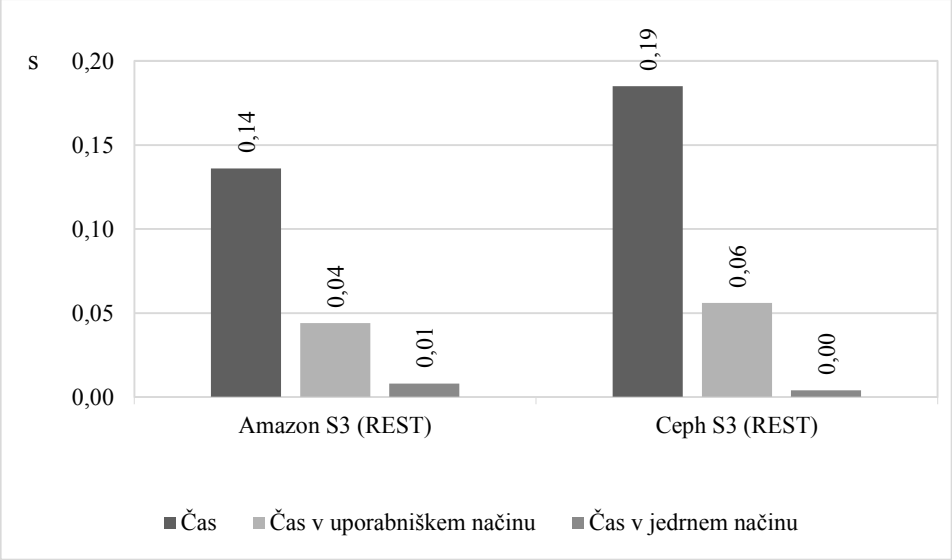
Slika 5.14: Čas branja (GET) velikega predmeta preko vmesnika REST.



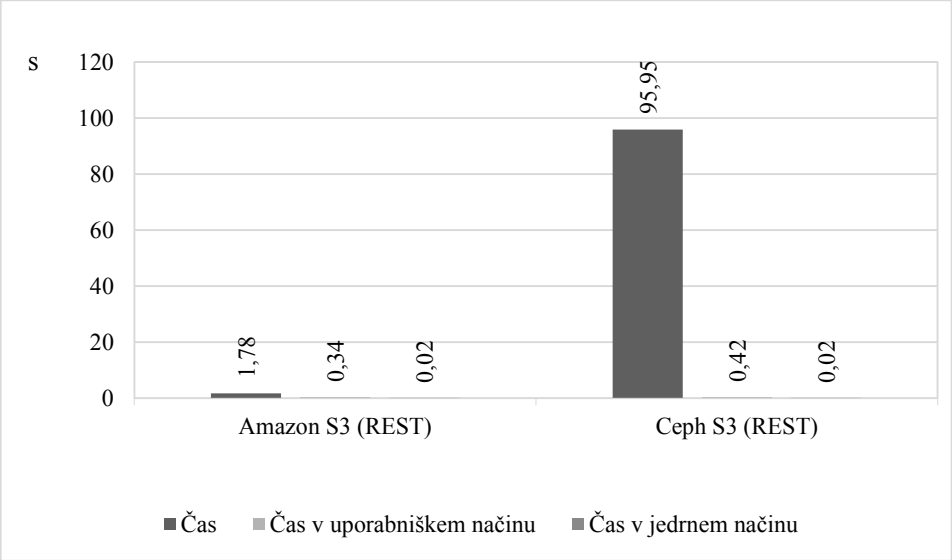
Slika 5.15: Čas branja (GET) veliko majhnih predmetov preko vmesnika REST.

5.2.3 Brisanje predmeta v shrambi S3

Primerjali smo čase brisanja predmetov v shrambi Amazon S3 in Ceph S3. Slika 5.16 prikazuje čas brisanja velikega predmeta v obeh shrambah, Slika 5.17 pa brisanje veliko majhnih predmetov. V prvem primeru sta se obe shrambi odrezali enako hitro, za izbris majhnih predmetov pa je shramba Ceph S3 potrebovala kar petintridesetkrat več časa kot Amazon S3.



Slika 5.16: Čas brisanja (DELETE) velikega predmeta preko vmesnika REST.



Slika 5.17: Čas brisanja (DELETE) več manjših predmetov preko vmesnika REST.

Poglavje 6 Zaključek

V diplomski nalogi smo podrobneje spoznali in opisali predmetno shrambo tako kot spletno storitev kot tudi način shranjevanja podatkov v porazdeljeni shrambi. Kot primer spletne predmetne shrambe smo navedli shrambo Amazon S3, kjer smo se spoznali s programskim vmesnikom REST preko protokola HTTP in z aplikacijami za dostop do shrambe S3. Za primer porazdeljene shrambe smo izbrali sistem Ceph, ki smo ga postavili na gručo strežnikov. Opisali smo sestavne dele sistema Ceph in možnosti dostopa preko različnih vmesnikov. Osredotočili smo se predvsem na prehod RADOS z vmesnikom S3 in datotečni sistem CephFS.

Med nameščanjem sistema Ceph in strežnikov NFS in SMB smo se podrobno spoznali z operacijskim sistemom Linux in načrtovanjem računalniškega omrežja z ločeno omrežno hrbtenico.

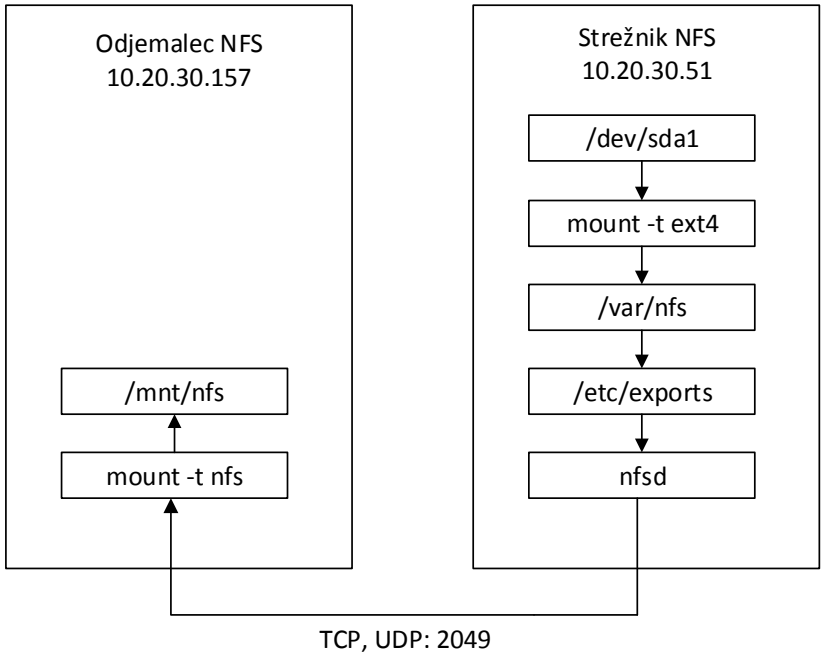
Primerjali smo dostop do shrambe Ceph S3 s plačljivo shrambo Amazon S3. Meritve so pokazale, da je shramba Ceph S3 primerljiva z Amazonovo shrambo. Ceph S3 bi lahko še dodatno optimizirali z uporabo drugega, hitrejšega spletnega strežnika in z uporabo izenačevalnikov obremenitve, ki bi poizvedbo preusmerili na enega izmed mnogih prehodov RADOS.

Datotečni sistem CephFS smo primerjali z najbolj razširjenima omrežnima protokoloma NFS in SMB. Čeprav sta strežnika NFS in SMB enostavnejša za postavitev in namestitev odjemalca, nudi CephFS večjo zanesljivost zaradi replikacije podatkov takoj ob zapisu in lahžjo razširljivost. Za povečanje skupne kapacitete shrambe lahko kadarkoli v sistem dodamo nov strežnik z OSDji, kar pri NFS in SMB ni mogoče. Izmerjena hitrost zapisovanja v CephFS je bila nižja zaradi replikacije ob zapisu. Tako hitrosti zapisovanja kot tudi hitrosti branja bi lahko izboljšali z uporabo hitrih diskov SSD v vsakem strežniku za dnevnik OSDjev oziroma predpomnilnik. Sistem Ceph omogoča visoke hitrosti branja, ker se vsak shranjen predmet nahaja na več shranjevalnih napravah, kar porazdeli branje med več naprav.

Poleg omenjenih načinov dostopa ponuja Ceph še možnost dostopa do bločne naprave (Ceph RBD), ki nudi bločno napravo za uporabo z navideznimi napravami v sistemih kot so QEMU, OpenStack in CloudStack. Prehod RADOS poleg vmesnika S3 nudi tudi vmesnik OpenStack

Swift. Navsezadnje pa lahko s programskim vmesnikom knjižnice librados implementiramo povsem svoj vmesnik do predmetne shrambe.

Dodatek A Namestitev strežnika in odjemalca NFS



Slika A.1: Diagram povezave med strežnikom in odjemalcem NFS. Na strežniku smo izvozili direktorij /var/nfs, ki smo ga na odjemalcu priključili v /mnt/nfs. Povezava poteka preko vrat 2049, preko protokolov TCP in UDP.

Slika A.1 prikazuje povezavo med strežnikom in odjemalcem NFS. Za namestitev strežnika NFS smo namestili ustrezne pakete, ustvarili imenik, ki ga bomo delili z ostalimi in storitev dodali v samodejni zagon ob vsakem zagonu računalnika.

1. Namestimo pakete: `sudo apt-get install nfs-kernel-server portmap nfs-common`
2. Ustvarimo nov imenik, ki ga bomo delili z ostalimi: `mkdir /var/nfs`
3. Dodamo vnos v datoteko `/etc/exports`:
   `/var/nfs 192.168.4.0/24(rw,async,no_subtree_check)`
4. Ponovno zaženemo storitev:
   `sudo /etc/init.d/nfd-kernel-server restart`

5. Nastavimo samodejni zagon storitve ob zagonu strežnika:
   `sudo update-rc.d rpcbind enable && sudo update-rc.d nfs-common enable`

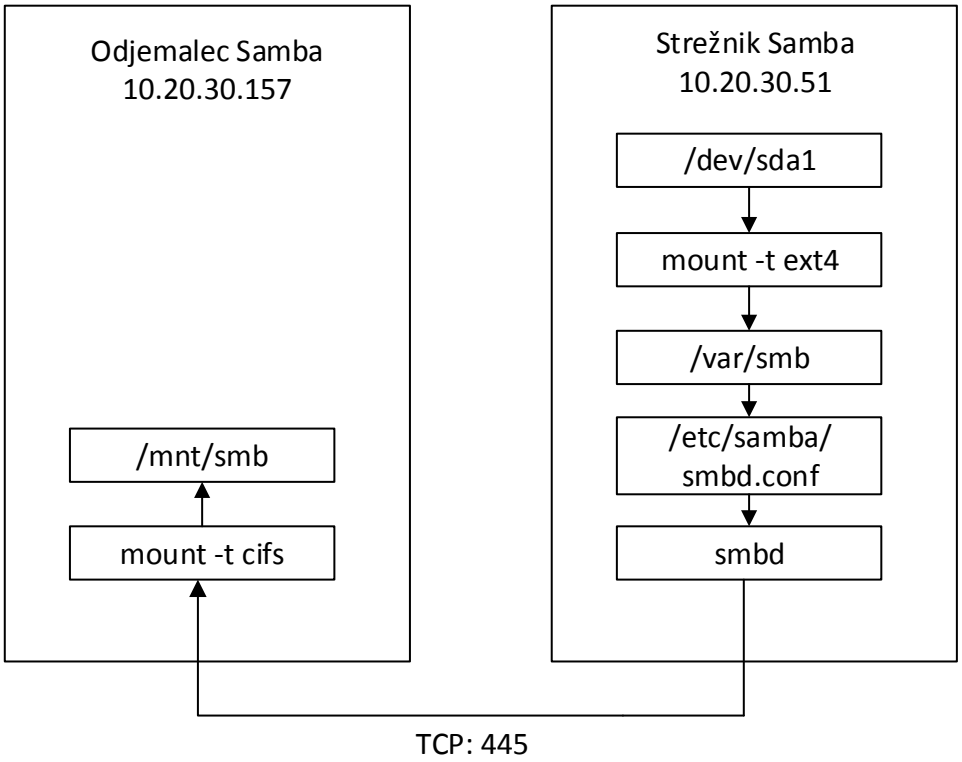
Odjemalca NFS smo namestili s paketom `nfs-common`, in ustvarili nov imenik, ki bo služil kot točka priklopa shrambe. Z ukazom mount smo priključili shrambo na odjemalca.

1. Namestimo pakete: `sudo apt-get install nfs-common`
2. Ustvarimo nov imenik za točko priklopa: `sudo mkdir /media/nfs`
3. Priključimo shrambo z ukazom `mount` (Izpis A.1).

```
jan@itpc:~$ sudo mount -t nfs 10.20.30.51:/var/nfs /mnt/nfs/ -v
mount.nfs: timeout set for Sun Feb 28 01:54:07 2016
mount.nfs: trying text-based options 'vers=4,addr=10.20.30.51,clientaddr=10.20.3
0.157'
```

Izpis A.1: Z ukazom `mount -t nfs` priključimo shrambo na odjemalca.

Dodatek B Namestitev strežnika in odjemalca Samba



Slika B.1: Diagram povezave med strežnikom in odjemalcem SMB. Na strežniku smo v nastavitveni datoteki nastavili direktorij /var/smb kot deljeno mapo. Na odjemalcu smo jo v priključili /mnt/smb. Povezava poteka preko vrat 445, preko protokola TCP.

Slika B.1 prikazuje diagram povezave med strežnikom in odjemalcem SMB. Za namestitev strežnika SMB smo namestili paket `samba` in ustvarili imenik, ki ga bomo delili z ostalimi.

1. Namestimo pakete: `sudo apt-get install samba`
2. Ustvarimo nov imenik, ki ga bomo delili z ostalimi: `mkdir /var/smb`
3. Spremenimo lastništvo imenika, da bo imenik brez lastnika:
   `sudo chown –R nobody.nogroup /var/smb`
4. Dodamo vnos v nastavitveno datoteko `/etc/samba/smb.conf`:

```
[shared]
path = /var/smb
```

```
browsable = yes
read only = no
guest ok = yes
```

5. Ponovno zaženemo storitev:
   `sudo service samba restart`

Odjemalca SMB smo namestili s paketom `cifs-utils`, in ustvarili nov imenik, ki bo služil kot točka priklopa shrambe. Nato smo z ukazom `mount` priključili shrambo na strežnik:

4. Namestimo pakete: `sudo apt-get install cifs-utils`
5. Ustvarimo nov imenik za točko priklopa: `sudo mkdir /media/smb`
6. Priključimo shrambo z ukazom `mount` in po potrebi vnesemo geslo (Izpis B.1).

```
jan@itpc:~$ sudo mount -t cifs //10.20.30.51/smb /mnt/smb -o user=jan,pass=opica
 -v
mount.cifs kernel mount options: ip=10.20.30.51,unc=\\10.20.30.51\smb,user=jan,p
ass=********
```

Izpis B.1: Z ukazom `mount -t cifs` priključimo shrambo na odjemalca.

Dodatek C Namestitev Ceph

V tem dodatku bomo opisali postopek namestitve porazdeljene gruče Ceph na štiri strežnike. Specifikacije strežnikov opisuje Tabela C.1.

Lastnosti	**Strežnik**
Operacijski sistem	Linux, Ubuntu Server 14.04
Centralna procesorska enota	intel Celeron N3150 @ 2,08GHz
Delovni pomnilnik	16 GB
Omrežna povezava	dve omrežni kartici hitrosti 1 GBit/s
Trdi disk	razni trdi diski (WD Black, WD Green) kapacitet med 500 GB in 2 TB

Tabela C.1: Specifikacije strežnikov v gruči Ceph.

Ubuntu Server 14.04 LTS

Na štiri strežnike smo namestili operacijski sistem Linux, distribucijo Ubuntu Server 14.04 LTS. Med namestitvijo smo že izbrali pakete za oddaljen dostop (SSH), po končani namestitvi pa smo za posodobitev sistema in nameščenih paketov izvedli ukaza `sudo apt-get update` in `sudo apt-get upgrade`.

Na strežnikih smo ustvarili novega uporabnika `ceph`, ki smo omogočili administratorski dostop `sudo`.

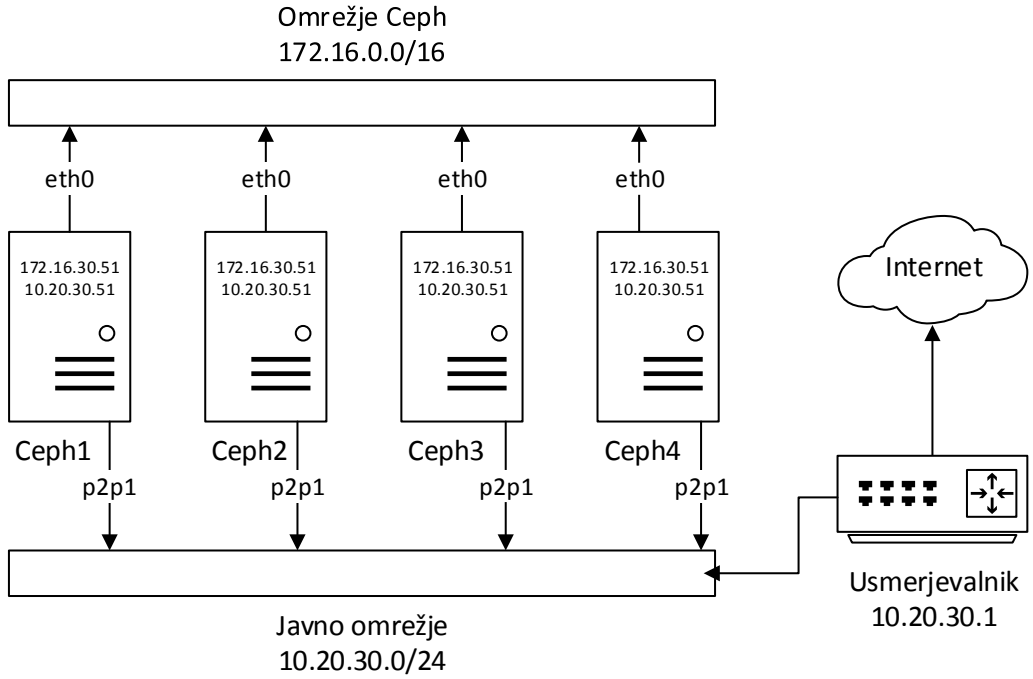
```
~$ sudo useradd -d /home/ceph -m ceph
~$ sudo passwd ceph
```

Vnos v datoteki `/etc/sudoers` za dostop `sudo`:

```
ceph ALL = (root) NOPASSWD:ALL
```

Omrežna povezljivost

Strežnike smo povezali v dve ločeni računalniški omrežji (Slika C.1). Zasebno omrežje Ceph je namenjeno izključno interni komunikaciji med strežniki za potrebe storitve Ceph, javno omrežje pa za dostop do spleta in oddaljeno upravljanje preko varne lupine (SSH).



Slika C.1: Omrežni diagram okolja Ceph. Strežniki so povezani v zasebno omrežje Ceph za potrebe komunikacije znotraj gruče in v javno omrežje z dostopom do spleta za dostop odjemalcev do gruče.

Določili smo statične naslove IP, naslov maske, prehoda in domenskega strežnika. Izpis C.1 prikazuje omrežne nastavitve v datoteki `/etc/network/interfaces` na strežniku ceph1. Na enak način smo omrežni naslov nastavili na ostalih strežnikih.

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface - on motherboard
auto p2p1
iface p2p1 inet static
        address 10.20.30.51
        netmask 255.255.255.0
        gateway 10.20.30.1
        dns-nameservers 10.20.30.1

# PCIe network interface
auto eth0
iface eth0 inet static
        address 172.16.30.51
        netmask 255.255.0.0
```

Izpis C.1: Omrežne nastavitve v datoteki `interfaces` na strežniku ceph1.

Ker na zasebnem omrežju nimamo nastavljenega domenskega strežnika, moramo imena strežnikov in njihove naslove IP zapisati neposredno v datoteko `/etc/hosts`. Znotraj zasebnega omrežja bomo strežnike naslavljali po kratkih imenih, za oddaljen dostop preko javnega omrežna pa s polnim domenskim imenom (FQDN). Vsebina datoteke na vsakem strežniku:

```
172.16.30.51    ceph1
172.16.30.52    ceph2
172.16.30.53    ceph3
172.16.30.54    ceph4

10.20.30.51     ceph1.ceph.lan
10.20.30.52     ceph2.ceph.lan
10.20.30.53     ceph3.ceph.lan
10.20.30.54     ceph4.ceph.lan
```

S temi nastavitvami bomo lahko strežnike naslavljali po imenih namesto po naslovih IP.

Varna lupina (SSH)

Dostop preko varne lupine bomo uporabljali za oddaljeno upravljanje s strežniki, in namestitev Cepha z uporabo programa ceph-deploy. V konfiguracijsko datoteko SSH smo dodali statične vnose, s katerimi definiramo privzeto uporabniško ime ter omrežni naslov za izbrana imena strežnikov:

```
Host ceph1
```

```
    Hostname ceph1
    User ceph
Host ceph2
    Hostname ceph2
    User ceph
Host ceph3
    Hostname ceph3
    User ceph
Host ceph4
    Hostname ceph4
    User ceph
```

Za nemoteno povezovanje med strežniki smo omogočili še prijavo s ključi namesto uporabe gesel. Na administratorskem strežniku smo ustvarili par zasebnega in javnega ključa z ukazom `ssh-keygen`, in javni ključ prenesemo na vse ostale strežnike v gruči:

```
ceph@ceph1:~$ ssh-copy-id ceph1
ceph@ceph1:~$ ssh-copy-id ceph2
ceph@ceph1:~$ ssh-copy-id ceph3
ceph@ceph1:~$ ssh-copy-id ceph4
```

Uporabniških imen nam ni potrebno navesti, ker smo jih že predhodno vnesli v konfiguracijsko datoteko. Na vsakega izmed naštetih strežnikov se prvič povežemo ročno, da preverimo ustreznost nastavitev in da sprejmemo prstni odtis (Izpis C.2) vsakega strežnika ter ga s tem dodamo na seznam znanih gostiteljev.

```
ceph@ceph1:~$ ssh ceph1
The authenticity of host 'ceph1 (127.0.1.1)' can't be established.
ECDSA key fingerprint is 1a:d3:6b:f5:24:9b:c8:36:7b:5e:24:d9:70:73:6d:b3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ceph1' (ECDSA) to the list of known hosts.
```

Izpis C.2: Sprejemanje prstnega odtisa ob prvem povezovanju preko SSH.

Sinhronizacija časa z omrežnim časovnim strežnikom

Monitorji, ki jih uporablja Ceph za nadzor sistema, so odvisni od točnega časa, zato smo na vse strežnike namestili paket ntp, ki skrbi za sinhronizacijo časa v ozadju.

```
~$ sudo apt-get install ntp
```

Po namestitvi v konfiguracijsko datoteko `/etc/ntp.conf` vnesemo naslove izbranih časovnih strežnikov:

```
server ntp1.arnes.si
server ntp2.arnes.si
server 0.ubuntu.pool.ntp.org
server 1.ubuntu.pool.ntp.org
server 2.ubuntu.pool.ntp.org
server 3.ubuntu.pool.ntp.org
```

Datoteko shranimo, in ponovno zaženemo proces `ntp`:

```
~$ sudo service ntp reload
```

Stanje sinhronizacije časa preverimo z ukazom `sudo ntpq -p` (Izpis C.3).

```
ceph@ceph1:~$ sudo ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+ntp3.arnes.si   .GPS.            1 u   25 1024  377   16.650    0.591   1.258
*planja.arnes.si .shm0.           1 u 1032 1024  377    9.865    3.344   2.388
 svr.irssi.si    12.1.72.21       2 u   30h 1024    0    1.833    2.186   0.000
-bsn-77-94-137-1 193.2.1.92       2 u  525 1024  377    3.257   -1.806  18.691
-2a01:260:4082:1 211.172.242.174  3 u  674 1024  177   28.085    0.910   0.895
-cad.lecad.fs.un 193.2.1.117      2 u  278 1024  377    1.239    3.049   0.336
+golem.canonical 193.79.237.14    2 u  407 1024  377   42.781    0.858   0.379
```

Izpis C.3: Stanje sinhronizacije časa s strežniki NTP:

Ceph-deploy

Ceph bomo namestili s programom ceph-deploy, s katerim bomo na strežnike nameščali Ceph in v gručo dodali vse strežnike. Paket ceph-deploy namestimo le na enem strežniku, ki ga bomo uporabili za namestitev na vse ostale strežnike. Temu strežniku bomo rekli administratorski strežnik, izbrali pa bomo strežnik ceph1. Nekateri moduli programa ceph-deploy so še v razvoju, zato smo jih kasneje izvedli ročno.

Preden smo lahko namestili ceph-deploy smo morali dodati ključ izdajatelja (Izpis C.4) in repozitorij Ceph za ustrezno distribucijo operacijskega sistema v lokalen seznam virov programa za upravljanje s paketi `apt` (Izpis C.5).

```
ceph@ceph1:~$ wget -4 https://download.ceph.com/keys/release.asc
--2016-01-14 14:21:31--  https://download.ceph.com/keys/release.asc
Resolving download.ceph.com (download.ceph.com)... 173.236.253.173
Connecting to download.ceph.com (download.ceph.com)|173.236.253.173|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 1645 (1.6K) [text/plain]
Saving to: 'release.asc.1'

100%[======================================>] 1,645       --.-K/s   in 0s

2016-01-14 14:21:32 (14.0 MB/s) - 'release.asc.1' saved [1645/1645]

ceph@ceph1:~$ sudo apt-key add release.asc
OK
```

Izpis C.4: Prenos in namestitev ključa izdajatelja.

```
 GNU nano 2.2.6       File: /etc/apt/sources.list.d/ceph.list          Modified

deb http://ceph.com/debian-hammer/ trusty main
```

Izpis C.5: Repozitorij Ceph dodamo v seznam virov apt.

Po dodanem ključu izdajatelja in dodanem repozirotiju moramo osvežiti vsebine paketov z ukazom `sudo apt-get update`, in nato namestimo paket ceph-deploy:

```
ceph@ceph1:~$ sudo apt-get install ceph-deploy
```

Ko na administratorski strežnik namestimo ceph-deploy, ustvarimo nov imenik, iz katerega ga bomo izvajali, saj ceph-deploy za svoje delovanje ustvari konfiguracijsko datoteko, dnevnik akcij in ključe, in pričakuje, da ga bomo vedno izvajali iz istega imenika.

```
ceph@ceph1:~$ mkdir cluster && cd cluster
```

S postopkom namestitve začnemo z izvedbo ukaza `ceph-deploy new`, ki zgoraj omenjene datoteke ustvari (Izpis C.6). Kot argument temu ukazu podamo seznam strežnikov, kjer bodo nameščeni monitorji.

```
ceph@ceph1:~/cluster$ ceph-deploy new ceph2 ceph3 ceph4
```

```
ceph@ceph1:~/ceph-cluster$ ll
total 348
drwxrwxr-x 2 ceph ceph   4096 Dec 27 16:57 ./
drwxr-xr-x 9 ceph ceph   4096 Jan 14 14:21 ../
-rw-rw-r-- 1 ceph ceph    629 Dec  6 11:09 ceph.conf
-rw-rw-r-- 1 ceph ceph 313063 Jan  1 13:49 ceph.log
-rw------- 1 ceph ceph     73 Sep 11 15:57 ceph.mon.keyring
ceph@ceph1:~/ceph-cluster$
```

Izpis C.6: Datoteke, ki se ustvarijo po izvedbi ukaza `ceph-deloy new`.

V konfiguracijsko datoteko (Izpis C.7) smo dodali vnosa za javno in zasebno omrežje. Konfiguracijska datoteka se ob prvi namestitvi samodejno prenese na vse strežnike, če pa jo urejamo kasneje, jo lahko na strežnike prenesemo z ukazom `ceph-deploy config push`.

```
[global]
fsid = 8c216865-f8b3-49ea-bf1f-e88ec7d84578
mon_initial_members = ceph2, ceph3, ceph4
mon_host = 172.16.30.52,172.16.30.53,172.16.30.54
auth_cluster_required = cephx
auth_service_required = cephx
auth_client_required = cephx
filestore_xattr_use_omap = true
;keyring = /etc/ceph/ceph.client.admin.keyring
public network = 10.20.30.0/24
cluster network = 172.16.0.0/16
```

Izpis C.7: Vsebina konfiguracijske datoteke `ceph.conf`.

Ceph namestimo z ukazom `ceph-deploy install ceph{1,2,3,4}` (Izpis C.8). S tem ukazom se `ceph-deploy` preko SSH poveže na vsakega izmed strežnikov, namesti ključ izdajatelja ter doda repozitorij Ceph v seznam virov za `apt-get`, kar smo predhodno storili na administratorskem strežniku za namestitev paketa `ceph-deploy`. Izvede se posodobitev paketov z ukazom `apt-get update` in na vsak strežnik se namestijo paketi `ceph`, `ceph-mds` in `radosgw`. Na strežnike, ki smo jih predhodno označili kot monitorje, se z ukazom `install` namestijo monitorji ceph-mon ki se dodajo v gručo.

```
ceph@ceph1:~/ceph-cluster$ ceph-deploy install ceph{1,2,3,4}
[ceph_deploy.conf][DEBUG ] found configuration file at: /home/ceph/.cephdeploy.conf
[ceph_deploy.cli][       ] Invoked (1.5.28): /usr/bin/ceph-deploy install ceph1 ceph2 ceph3 ceph4
[ceph_deploy.cli][       ] ceph-deploy options:
[ceph_deploy.cli][       ]  verbose                       : False
[ceph_deploy.cli][       ]  testing                       : None
[ceph_deploy.cli][       ]  cd_conf                       : <ceph_deploy.conf.cephdeploy.Conf instanc
e at 0x7fb1662d3710>
[ceph_deploy.cli][       ]  cluster                       : ceph
[ceph_deploy.cli][       ]  install_mds                   : False
[ceph_deploy.cli][       ]  stable                        : None
[ceph_deploy.cli][       ]  default_release               : False
[ceph_deploy.cli][       ]  username                      : None
[ceph_deploy.cli][       ]  adjust_repos                  : True
[ceph_deploy.cli][       ]  func                          : <function install at 0x7fb166b95de8>
[ceph_deploy.cli][       ]  install_all                   : False
[ceph_deploy.cli][       ]  repo                          : False
[ceph_deploy.cli][       ]  host                          : ['ceph1', 'ceph2', 'ceph3', 'ceph4']
[ceph_deploy.cli][       ]  install_rgw                   : False
[ceph_deploy.cli][       ]  repo_url                      : None
[ceph_deploy.cli][       ]  ceph_conf                     : None
[ceph_deploy.cli][       ]  install_osd                   : False
[ceph_deploy.cli][       ]  version_kind                  : stable
[ceph_deploy.cli][       ]  install_common                : False
[ceph_deploy.cli][       ]  overwrite_conf                : False
[ceph_deploy.cli][       ]  quiet                         : False
[ceph_deploy.cli][       ]  dev                           : master
[ceph_deploy.cli][       ]  local_mirror                  : None
[ceph_deploy.cli][       ]  release                       : None
[ceph_deploy.cli][       ]  install_mon                   : False
[ceph_deploy.cli][       ]  gpg_url                       : None
[ceph_deploy.install][DEBUG ] Installing stable version hammer on cluster ceph hosts ceph1 ceph2 cep
h3 ceph4
```

Izpis C.8: Začetek namestitve Cepha na vse strežnike z ukazom `ceph-deploy install`.

Ceph-OSD

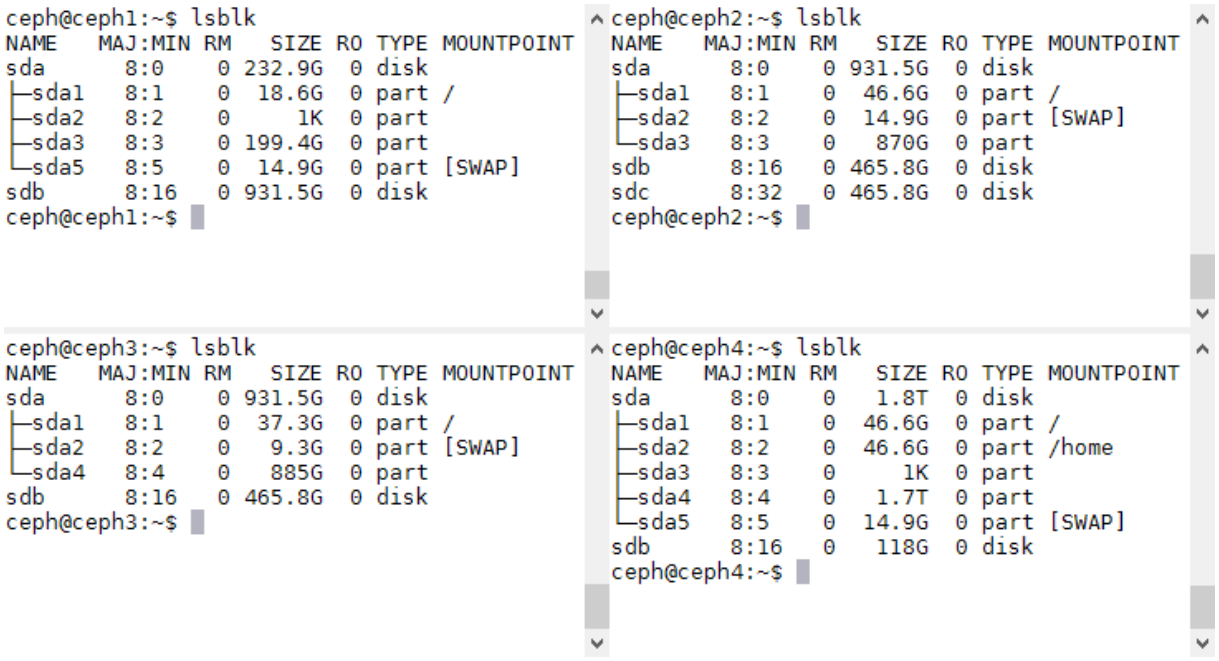
Po uspešni namestitvi Cepha in monitorjev lahko v gručo dodamo pomnilniške naprave OSD. Za vsak OSD skrbi proces ceph-osd, ki predmete shranjuje na trdi disk. Preden OSDje dodamo v gručo, najprej identificiramo vse trde diske, ki so na strežnike priključeni. Ceph-deploy omogoča izpis vseh diskov na vseh strežnikih v gruči z ukazom `disk list`, ta ukaz nam vrne izpis diskov ter particij, datotečnih sistemov particij ter pot na katero je particija priključena (Izpis C.9), ne pove pa nam same velikosti diska oziroma particije:

```
ceph@ceph1:~/ceph-cluster$ ceph-deploy disk list ceph{1,2,3,4}
```

```
[ceph1][      ] Running command: sudo /usr/sbin/ceph-disk list
[ceph1][DEBUG ] /dev/sda :
[ceph1][DEBUG ]  /dev/sda1 other, ext4, mounted on /
[ceph1][DEBUG ]  /dev/sda2 other, 0x5
[ceph1][DEBUG ]  /dev/sda3 other, xfs
[ceph1][DEBUG ]  /dev/sda5 swap, swap
[ceph1][DEBUG ] /dev/sdb other, unknown

[ceph2][      ] Running command: sudo /usr/sbin/ceph-disk list
[ceph2][DEBUG ] /dev/sda :
[ceph2][DEBUG ]  /dev/sda1 other, ext4, mounted on /
[ceph2][DEBUG ]  /dev/sda2 swap, swap
[ceph2][DEBUG ]  /dev/sda3 other, xfs
[ceph2][DEBUG ] /dev/sdb other, unknown
[ceph2][DEBUG ] /dev/sdc other, unknown

[ceph3][      ] Running command: sudo /usr/sbin/ceph-disk list
[ceph3][DEBUG ] /dev/sda :
[ceph3][DEBUG ]  /dev/sda1 other, ext4, mounted on /
[ceph3][DEBUG ]  /dev/sda2 swap, swap
[ceph3][DEBUG ]  /dev/sda4 other, xfs
[ceph3][DEBUG ] /dev/sdb other, unknown

[ceph4][      ] Running command: sudo /usr/sbin/ceph-disk list
[ceph4][DEBUG ] /dev/sda :
[ceph4][DEBUG ]  /dev/sda1 other, ext4, mounted on /
[ceph4][DEBUG ]  /dev/sda2 other, ext4, mounted on /home
[ceph4][DEBUG ]  /dev/sda3 other, 0x5
[ceph4][DEBUG ]  /dev/sda4 other, xfs
[ceph4][DEBUG ]  /dev/sda5 swap, swap
[ceph4][DEBUG ] /dev/sdb other, unknown
```

Izpis C.9: Izpis trdih diskov z ukazom `ceph-deploy disk list`.

Informacije o diskih in particijah v Linuxu vrne ukaz `lsblk`. Ta ukaz nam za razliko od `ceph-deploy disk list` pove še kapaciteto, vendar ne izpiše datotečnega sistema particije. Preko protokola SSH se povežemo se na vsakega izmed strežnikov in ukaz izvedemo. Izpis C.10 prikazuje izpis ukaza `lsblk`.



Izpis C.10: Izpis priključenih pomnilniških naprav in particij z ukazom `lsblk`.

Z uporabo `ceph-deploy osd` dodamo v gručo bodisi prazne, neformatirane pomnilniške naprave, bodisi obstoječe particije, ki jih ustrezno formatiramo. Particije, ki jih bomo uporabili, moramo najprej odklopiti iz sistema z ukazom `umount`. Če obstajajo zapisi za samodejno priključitev particij ob zagonu sistema v datoteki `/etc/fstab`, jih moramo odstraniti.

Izbrali smo pomnilniške naprave, ki jih bomo dodali v gruči. Na strežnikih ceph1, ceph3 in ceph4 bomo za OSD uporabili trde diske `sdb`, na strežniku ceph2 pa diska `sdb` in `sdc`.

Z ukazom `disk zap` zbrišemo morebitno obstoječo particijsko tabelo na trdem disku za nadaljnjo uporabo v gruči Ceph:

```
ceph@ceph1:/ceph-cluster~$ ceph-deploy disk zap ceph1:sdb
ceph2:sdb ceph2:sdc ceph3:sdb ceph4:sdb
```

Z ukazom `osd prepare` ceph-deploy diske formatiramo v ustrezen datotečni sistem za uporabo v gruči Ceph. V tem ukazu lahko specificiramo ločeno pomnilniško napravo (na primer hiter disk SSD) kot disk za dnevnik (angl. *journal*), če tega ne storimo, bo `ceph-deploy` na pomnilniški napravi samodejno ustvaril ločeno particijo za uporabo dnevnika, če pa za OSD izberemo že obstoječo particijo brez ločene particijo za dnevnik, se le ta ustvari kot binarna datoteka. Izpis C.11 prikazuje strukturo imenika posamezne naprave OSD. Iz slike razberemo, da je datoteka `journal` bližnjica do particije na disku. V primeru, da dnevniku ne bi namenili

lastne particije, se dnevnik shranjuje kar v datoteko `journal`. Izvedemo ukaz `ceph-deploy osd prepare ceph1:sdb ceph2:sdb ceph2:sdc ceph3:sdb ceph4:sdb`.

Po formatiranju pomnilniških naprav particije aktiviramo kot naprave OSD z ukazom `osd activate`, s čimer se dodajo v gručo. Ukazu kot argumente podamo ime particije (`sda1`) in ne imen naprave (`sdb`):

```
~/ceph-cluster$ ceph-deploy osd activate ceph1:sdb1 ceph2:sdb1
ceph2:sdc1 ceph3:sdb1 ceph4:sdb1
```

```
ceph@ceph1:/var/lib/ceph/osd/ceph-3$ ll -h
total 68K
drwxr-xr-x   3 root root  217 Dec  6 19:31 ./
drwxr-xr-x   3 root root 4.0K Dec 31 15:20 ../
-rw-r--r--   1 root root  481 Nov 23 10:46 activate.monmap
-rw-r--r--   1 root root    3 Nov 23 10:46 active
-rw-r--r--   1 root root   37 Nov 23 10:46 ceph_fsid
drwxr-xr-x 336 root root  16K Dec 14 22:10 current/
-rw-r--r--   1 root root   37 Nov 23 10:46 fsid
lrwxrwxrwx   1 root root   58 Nov 23 10:46 journal -> /dev/disk/
by-partuuid/4f745d2f-b6ee-452d-b9f3-d749990012c8
-rw-r--r--   1 root root   37 Nov 23 10:46 journal_uuid
-rw-------   1 root root   56 Nov 23 10:46 keyring
-rw-r--r--   1 root root   21 Nov 23 10:46 magic
-rw-r--r--   1 root root    6 Nov 23 10:46 ready
-rw-r--r--   1 root root    4 Nov 23 10:46 store_version
-rw-r--r--   1 root root   53 Nov 23 10:46 superblock
-rw-r--r--   1 root root    0 Dec  6 19:32 upstart
-rw-r--r--   1 root root    2 Nov 23 10:46 whoami
```

Izpis C.11: Sktruktura imenika OSD.

Ceph-MDS

Strežnik za meta podatke v primeru uporabe datotečnega sistema CephFS namestimo na strežnik z ukazom `ceph-deploy mds create`. MDS smo namestili na strežnik ceph1. Po namestitvi lahko stanje preverimo z ukazom `ceph mds stat` (Izpis C.12), iz katerega razberemo da je strežnik za meta podatke aktiven na strežniku ceph1.

```
ceph@ceph1:~$ ceph mds stat
e77: 1/1/1 up {0=ceph1=up:active}
```

Izpis C.12: Preverimo stanje MDS z ukazom `ceph mds stat`.

Datotečni sistem CephFS

Ko je strežnik za meta podatke postavljen, lahko namestimo datotečni sistem CephFS. CephFS omogočimo v dveh korakih. Najprej ustvarimo dva bazena, kamor bo datotečni sistem

shranjeval podatke in meta podatke, nato ustvarimo nov datotečni sistem. Bazenoma določimo število postavitvenih skupin. Te ukaze izvedemo na administratorskem strežniku, kjer imamo pravice za izvajanje ukaza `ceph`, in preverimo, če se je datotečni sistem ustvaril (Izpis C.13).

```
ceph@ceph1:~$ ceph osd pool create cephfs_data 128
ceph@ceph1:~$ ceph osd pool create cephfs_metadata 64
ceph@ceph1:~$ ceph fs new cephfs cephfs_metadata cephfs_data
```

```
ceph@ceph1:~$ ceph fs ls
name: cephfs, metadata pool: cephfs_metadata, data pools: [cephfs_data ]
ceph@ceph1:~$
```

Izpis C.13: Datotečni sistem izpišemo z ukazom `ceph fs ls`.

Na odjemalcu datotečni sistem CephFS priključimo s paketom `ceph-client`, ki v jedro namesti gonilnik za CephFS. Iz administratorskega strežnika pridobimo ključ, in ga skopiramo na odjemalca. Datotečni sistem priključimo z ukazom `mount -t ceph` ali kot datotečni sistem v uporabniškem načinu `ceph-fuse`.

```
sudo mount -t ceph 172.16.30.52:6789:/ /mnt/cephfs -o
name=admin,secret=AQAj3/JVY1kdLxAAn6EMh8CytPTlwfabTkWESw==

ceph-fuse -m 172.16.30.52:6789 /mnt/cephfs-fuse
```

Prehod RADOS

Na strežnikih, ki jih želimo uporabiti kot prehod RADOS, moramo imeti nameščen paket `radosgw`, ki se v primeru namestitve Cepha s ceph-deploy namesti samodejno.

Vsaka instanca prehoda RADOS potrebuje prijavne podatke za komunikacijo z gručo Ceph. Na administratorskem strežniku ustvarimo novo hrambo ključev (angl. *keyring*) in ustvarimo novo uporabniško ime in ključ za odjemalca. Odjemalčev ključ dodamo v gručo Ceph, in hrambo ključev distribuiramo na strežnik, kjer bo nameščen prehod RADOS. Nov ključ ustvarimo z ukazom ceph-authtool in ustvarjeni datoteki dodamo pravice za branje (Izpis C.14).

```
ceph@ceph1:~$ sudo ceph-authtool --create-keyring /etc/ceph/ceph.client.radosgw.
keyring
creating /etc/ceph/ceph.client.radosgw.keyring
ceph@ceph1:~$ sudo chmod +r /etc/ceph/ceph.client.radosgw.keyring
ceph@ceph1:~$
```

Izpis C.14: Ustvarimo novo hrambo ključev.

Vsakemu odjemalcu ustvarimo uporabniško ime (client.radosgw.gateway) in ključ za dostop ter mu dodamo pravice za branje in pisanje:

```
sudo ceph-authtool /etc/ceph/ceph.client.radosgw.keyring -n client.radosgw.gateway --gen-key
```

```
sudo ceph-authtool -n client.radosgw.gateway --cap osd 'allow rwx' --cap mon 'allow rwx' /etc/ceph/ceph.client.radosgw.keyring
```

Datoteko s ključi moramo prenesti na vse strežnike, kjer bi želeli namestiti prehod RADOS.

Prehod RADOS podatke shranjuje v več bazenov, ki se ustvarijo sami, če uporabniškem imenu nastavimo pravice za pisanje, sicer moramo bazene ustvariti ročno. Bazeni, ki se ustvarijo imajo predpono `.rgw`. Seznam bazenov prikazuje Izpis C.15.

```
ceph@ceph1:~$ rados lspools | grep .rgw
.rgw.root
.rgw.control
.rgw
.rgw.gc
.rgw.buckets.index
.rgw.buckets
.rgw.buckets.extra
ceph@ceph1:~$
```

Izpis C.15: Seznam bazenov, ki jih ustvari prehod RADOS.

Bazene lahko dodamo ročno z ukazom `ceph osd pool create`, ki mu z argumenti podamo želeno ime bazena in število postavitvenih skupin.

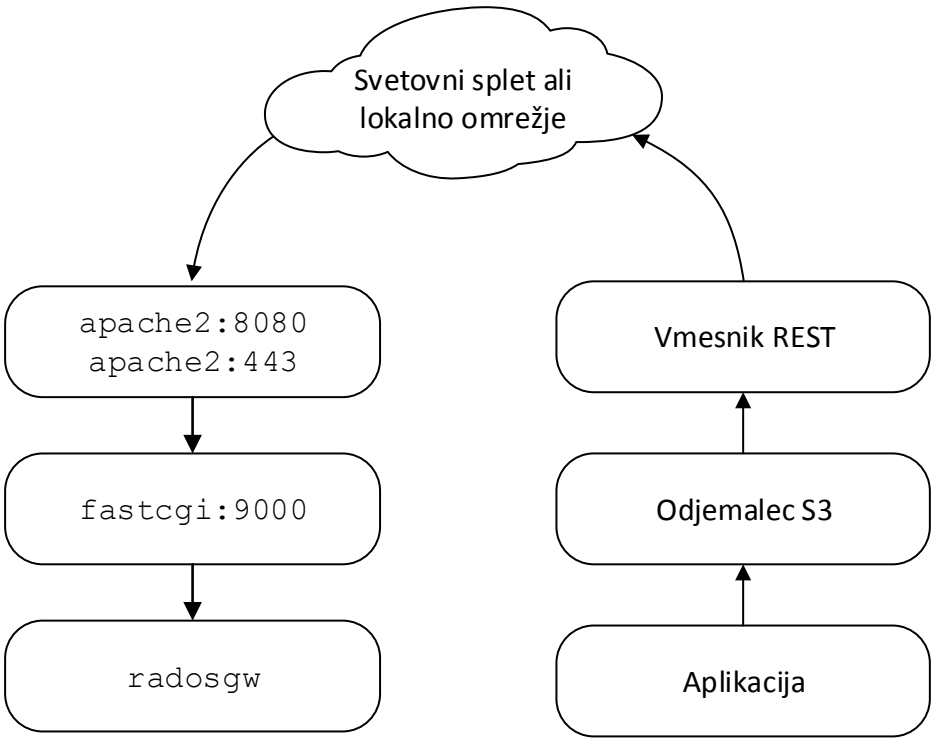
Konfiguracijo prehoda moramo shraniti v obstoječo nastavitveno datoteko ceph.conf in nove nastavitve prenesti na vse strežnike v gruči. Izpis C.16 prikazuje novo namestitveno datoteko z dodanimi nastavitvami za prehod RADOS.

```
 GNU nano 2.2.6                File: ceph.conf

[global]
fsid = 8c216865-f8b3-49ea-bf1f-e88ec7d84578
mon_initial_members = ceph2, ceph3, ceph4
mon_host = 172.20.30.52,172.20.30.53,172.20.30.54
auth_cluster_required = cephx
auth_service_required = cephx
auth_client_required = cephx
filestore_xattr_use_omap = true
;keyring = /etc/ceph/ceph.client.admin.keyring
public network = 10.20.30.0/24
cluster network = 172.20.30.0/24


[client.radosgw.gateway]
host = ceph1
keyring = /etc/ceph/ceph.client.radosgw.keyring
rgw socket path = ""
log file = /var/log/radosgw/client.radosgw.gateway.log
rgw frontends = fastcgi socket_port=9000 socket_host=0.0.0.0
rgw print continue = false
```

Izpis C.16: Nastavitvena datoteka ceph.conf z dodanimi nastavitvami prehoda RADOS.

Prehod RADOS zaženemo z ukazom `radosgw start`, ki ob pravilnosti vseh nastavitev vrne rezultat `/usr/bin/radosgw is running`.

Spletni strežnik Apache



Slika C.2: Dostop do prehoda Rados preko spletnega strežnika Apache in modula fastcgi.

Spletni strežnik Apache [21] služi kot vmesnik med spletom in prehodom RADOS. Namestimo ga z ukazon `apt-get install apache2`.

Konfiguracijo pripravimo tako, da ima Apache vlogo posredniškega strežnika (angl. *proxy server*). Apache posluša na vratih 8080 za nešifrirano povezavo (*http*) in na vratih 443 za šifrirano povezavo (*https*), in promet posreduje procesu `fastcgi`, ki posluša na vratih 9000. Potek prometa prikazuje Slika C.2. Za nastavitve nešifrirane povezave ustvarimo in uredimo datoteko `/etc/apache2/sites-available/rgw.conf`, za šifrirano povezavo pa datoteko `default-ssl.conf`, ki se nahaja v istem imeniku.

```
 GNU nano 2.2.6                File: rgw.conf

<VirtualHost *:8080>

ServerName localhost
ServerAlias ceph1

DocumentRoot /var/www/html

ErrorLog /var/log/apache2/rgw_error.log
CustomLog /var/log/apache2/rgw_access.log combined

# LogLevel debug

RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization},L]
SetEnv proxy-nokeepalive 1
ProxyPass / fcgi://localhost:9000/

</VirtualHost>
```

Izpis C.17: Konfiguracijska datoteka `rgw.conf`.

Konfiguracijo strežnika Apache za nešifriran promet prikazuje Izpis C.17. Za šifriran promet je konfiguracija podobna, le da namesto vrat `8080` uporabimo vrata `443`. Za šifriran dostop smo omogočili modul šifrirane povezave z ukazom `a2enmod ssl` in generirali lastnoročno podpisan certifikat:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

Pot do certifikatov vnesemo v konfiguracijsko datoteko za šifriran promet:

```
SSLCertificateFile /etc/apache2/ssl/apache.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

Modul `fastcgi` moramo pred uporabo omogočiti z ukazom `a2enmod`. Ko modul omogočimo, proces `apache2` ponovno zaženemo (Izpis C.18), s čimer aktiviramo modul in novo konfiguracijo.

```
root@ceph1:~# a2enmod fastcgi
Enabling module fastcgi.
To activate the new configuration, you need to run:
  service apache2 restart
root@ceph1:~# service apache2 restart
 * Restarting web server apache2
root@ceph1:~#
```

Izpis C.18: Omogočimo modul fastcgi.

Vmesnik REST

Za uporabo vmesnika REST, kompatibilnega z S3, ustvarimo novega uporabnika in ključe za dostop z ukazom radosgw-admin [13]:

```
sudo radosgw-admin user create --uid="jan" --display-name="Jan Tomsic"
```

Rezultat ukaza (Izpis C.20) vrne podatke o novo ustvarjenem uporabniku in par ključev za dostop. Par generiranih ključev je enake oblike kot ključi, ki bi jih generiral Amazon AWS za dostop do shrambe Amazon S3. Ključa, ki smo jih dobili:

- Ključ za dostop (access_key): YFW5D9P1DXYN6KNY6PAP
- Skrivni ključ (secret_key): 65Q0UWS+eK+8SCbaeAWMndyypi\/78TUjDz5B2txf

Poleg ključev se izpišejo še pravice uporabnika ter morebitne omejitve količine podatkov, ki jih lahko uporabnik shrani.

Za testiranje dostopa do shrambe bomo uporabili knjižnico `boto` [6] in programski jezik Python [22]. Knjižnico namestimo z ukazom `sudo apt-get install python-boto`.

S kratko skripto [13] (Izpis C.21), ki se poveže na prehod, preverimo pravilnost konfiguracije radosgw, apache2 in uporabniških ključev za dostop. Skripta ustvari novo vedro z imenom »novo-vedro« in izpiše vsa vedra, vključno z ravno ustvarjenim, ki pripadajo uporabniku (Izpis C.19).

```
ceph@ceph1:~/00other$ python s3test.py
novo-vedro      2016-01-16T15:36:46.000Z
```

Izpis C.19: Izpis veder, ki jih vrne skripta.

```
ceph@ceph1:~$ sudo radosgw-admin user create --uid="jan" --display-
name="Jan Tomsic"
{
    "user_id": "jan",
    "display_name": "Jan Tomsic",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [],
    "keys": [
        {
            "user": "jan",
            "access_key": "YFW5D9P1DXYN6KNY6PAP",
            "secret_key": "65Q0UWS+eK+8SCbaeAWMndyypi\/78TUjDz5B2txf"
        }
    ],
    "swift_keys": [],
    "caps": [],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
    "temp_url_keys": []
}
```

Izpis C.20: Podatki o novo ustvarjenem uporabniku prehoda RADOS.

```
ceph@ceph1:~/00other$ cat s3test.py
import boto
import boto.s3.connection
access_key = 'YFW5D9P1DXYN6KNY6PAP'
secret_key = '65Q0UWS+eK+8SCbaeAWMndyypi/78TUjDz5B2txf'
conn = boto.connect_s3(
aws_access_key_id = access_key,
aws_secret_access_key = secret_key,
host = 'ceph1',
is_secure=True,
calling_format = boto.s3.connection.OrdinaryCallingFormat(),
)
bucket = conn.create_bucket('novo-vedro')
for bucket in conn.get_all_buckets():
        print "{name}\t{created}".format(
                name = bucket.name,
                created = bucket.creation_date,
)
```

Izpis C.21: Skripta, s katero preverimo dostop preko vmesnika S3.

Nadzor

Ob namestitvi in med njo moramo večkrat preveriti stanje gruče. Preverimo, če so diski uspešno dodani kot naprave OSD in če se je kapaciteta gruče ustrezno povečala, če so vsi monitorji dosegljivi in časovno sinhronizirani, če je na voljo dovolj postavitvenih skupin, ali če se je morda zgodila kakšna druga napaka. Ceph nudi lastne ukaze za pregled stanja, vendar smo za lažjo preglednost strežnike dodali v nadzorni sistem Zabbix. Za nadzor gruče Ceph smo pripravili nekaj lastnih parametrov, s katerimi bomo nadzirali stanje.

Najbolj osnoven ukaz, ki pove stanje gruče, je ukaz `ceph health` (Izpis C.22).

```
ceph@ceph1:~/ceph-cluster$ ceph health
HEALTH_OK
```

Izpis C.22: Izpis ukaza `ceph health`.

Ceph health vrne enostaven rezultat HEALTH_OK, v primeru da je vse v redu, ali enostavno sporočilo če je v gruči težava. Nekatere napake, na katere smo naleteli:

- HEALTH_WARN pool .rgw.buckets has too few pgs
- HEALTH_WARN too many PGs per OSD (414 > max 300)
- HEALTH_WARN mds cluster is degraded
- HEALTH_WARN mds ceph1 is laggy
- HEALTH_WARN clock skew detected on mon.ceph4; Monitor clock skew detected

- HEALTH_WARN 161 pgs degraded; 161 pgs stuck unclean; 161 pgs undersized; recovery 3311/40575 objects degraded (8.160%);
- HEALTH_WARN 59 pgs backfill; 14 pgs backfilling; 74 pgs stuck unclean; recovery 27574/55037 objects misplaced (50.101%)

Za bolj podroben izpis stanja uporabimo ukaz `ceph status` (Izpis C.23).

```
ceph@ceph1:~/ceph-cluster$ ceph status
    cluster 8c216865-f8b3-49ea-bf1f-e88ec7d84578
     health HEALTH_OK
     monmap e1: 3 mons at {ceph2=172.20.30.52:6789/0,ceph3=172.20.30.53:6789/0,c
eph4=172.20.30.54:6789/0}
            election epoch 100, quorum 0,1,2 ceph2,ceph3,ceph4
     mdsmap e77: 1/1/1 up {0=ceph1=up:active}
     osdmap e569: 7 osds: 7 up, 7 in
      pgmap v149015: 528 pgs, 14 pools, 63665 MB data, 37005 objects
            202 GB used, 5265 GB / 5468 GB avail
                 528 active+clean
```

Izpis C.23: Izpis ukaza `ceph status`.

Ceph status izpiše še dodatne podrobnosti trenutnega stanja. Vrstica `cluster` pomeni identifikator datotečnega sistema (FSID), vrstica `monmap` izpiše stanje monitorjev, v tej vrstici bi se izpisale morebitne težave s sinhronizacijo ure med monitorji.

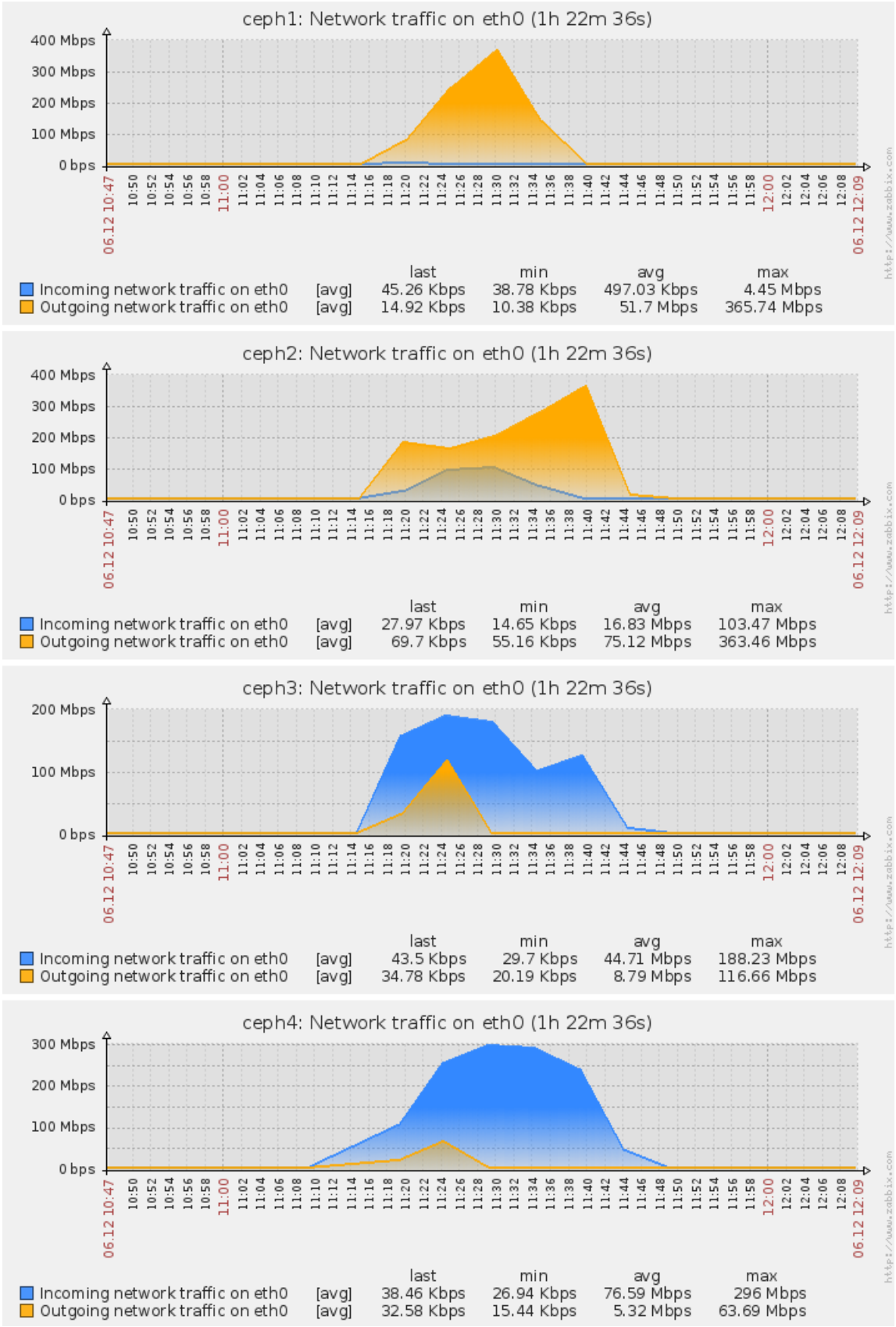
Vse strežnike smo dodali v nadzorni sistem Zabbix [23] [24]. Na strežnike smo namestili paket `zabbix-agent`, in na administratorskem strežniku ceph1 dodali lastne parametre za nadzor gruče Ceph. S takim nadzorom pridobimo večjo kontrolo nad gručo in posameznimi strežniki, saj se v Zabbixu vsa zgodovina beleži, ob nesreči pa nas sistem lahko opozori preko elektronske pošte ali sporočila SMS.

Zabbix-agent brez dodatnih parametrov nudi nadzor nad osnovnimi funkcijami strežnika, kot so poraba prostora na trdih diskih (Slika C.3), promet na mrežni povezavi, zasedenost pomnilnika RAM, trajanje neprekinjenega delovanja sistema itd.

Z opazovanjem podatkov nadzora smo med drugim lahko opazili, da se ob vsakem dodanem OSDju podatki v gruči uravnotežijo po vseh OSDjih, zaradi česar naraste promet na omrežni povezavi vseh strežnikov (Slika C.4). Ob nalaganju podatkov v shrambo preko prehoda RADOS na strežniku `ceph1`, je bila aktivnost pričakovano vidna na javnem omrežju le na tem strežniku, aktivnost na lokalnem omrežju gruče pa je narasla na vseh strežnikih.

Host	Name	Last value
ceph1	**Filesystems**	
	Free inodes on /var/lib/ceph/osd/ceph-3 (percentage)	99.97 %
	Free disk space on /var/lib/ceph/osd/ceph-3 (percentage)	95.02 %
ceph2	**Filesystems**	
	Free inodes on /var/lib/ceph/osd/ceph-0 (percentage)	99.97 %
	Free disk space on /var/lib/ceph/osd/ceph-0 (percentage)	95.2 %
	Free inodes on /var/lib/ceph/osd/ceph-6 (percentage)	99.98 %
	Free disk space on /var/lib/ceph/osd/ceph-6 (percentage)	95.55 %
ceph3	**Filesystems**	
	Free inodes on /var/lib/ceph/osd/ceph-1 (percentage)	99.97 %
	Free disk space on /var/lib/ceph/osd/ceph-1 (percentage)	95.4 %
	Free inodes on /var/lib/ceph/osd/ceph-4 (percentage)	99.97 %
	Free disk space on /var/lib/ceph/osd/ceph-4 (percentage)	95.17 %
ceph4	**Filesystems**	
	Free inodes on /var/lib/ceph/osd/ceph-2 (percentage)	99.98 %
	Free disk space on /var/lib/ceph/osd/ceph-2 (percentage)	97.3 %
	Free inodes on /var/lib/ceph/osd/ceph-5 (percentage)	99.89 %
	Free disk space on /var/lib/ceph/osd/ceph-5 (percentage)	96.3 %

Slika C.3: Delež prostega pomnilnika na posameznih OSDjih.



Slika C.4: Narast omrežnega prometa ob uravnoteženju gruče.

Poleg osnovnih funkcij smo za administratorski strežnik Ceph1 pripravili nekaj dodatnih parametrov [25], s katerimi nadzorujemo količino podatkov shranjenih v gruči, količino shranjenih predmetov, prostor, ki je v gruči še na voljo, število v gručo dodanih OSDjev, število postavitvenih skupin itd. Te parametre smo pripravili kot ukaze, ki iz ukazov `ceph_health`, `ceph_df` in `ceph_status` s pomočjo orodij `grep`, `cut` in `tr` ter z regularnimi izrazi za izbor števil izluščijo želene podatke (Izpis C.24). Slika C.5 prikazuje vrednosti, ki jih Zabbix pridobi z uporabo teh lastnih parametrov.

```
ceph@ceph1:/etc/zabbix/zabbix_agentd.d$ cat ceph.conf
# Ceph health
UserParameter=ceph_health,sudo ceph health

# Ceph df (total_bytes, total_avail_byted, total_used_bytes)
UserParameter=ceph df[*],sudo ceph df --format json-pretty | grep "$1" |
 grep -Eo '[0-9]{1,}'

# Number of pools
UserParameter=ceph_pools,sudo ceph status | grep pgmap | tr -d '[[:space:]]' |
 cut -d ',' -f 2 | grep -Eo '[0-9]*'

# Data stored in bytes
UserParameter=ceph_data,sudo ceph status -f json-pretty | grep data_bytes |
 grep -Eo '[0-9]*'

# Objects stored
UserParameter=ceph_objects,sudo ceph status | grep pgmap | tr -d '[[:space:]]'
| cut -d ',' -f 4 | grep -Eo '[0-9]*'

# Number of placement groups
UserParameter=ceph_pgs,sudo ceph status | grep pgmap | tr -d '[[:space:]]' |
 cut -d ',' -f 1 | cut -d ':' -f 2 | grep -Eo '[0-9]*'

# pgmap version
UserParameter=ceph_pgmap_version,sudo ceph status -f json-pretty | grep version
| grep -Eo '[0-9]*'

# Number of OSDs total
UserParameter=ceph_num_osds,sudo ceph status -f json-pretty | grep num_osds |
 grep -Eo '[0-9]*'

# Number of OSDs up
UserParameter=ceph_num_up_osds,sudo ceph status -f json-pretty |
 grep num_up_osds | grep -Eo '[0-9]*'

# Number of OSDs in
UserParameter=ceph_num_in_osds,sudo ceph status -f json-pretty |
 grep num_in_osds | grep -Eo '[0-9]*'
```

Izpis C.24: Parametri za nadzor gruče Ceph z sistemom Zabbix.

Name	Last check	Last value	Change
Ceph (1 Item)			
Ceph Health	2016-01-16 18:17:53	HEALTH_OK	-
Ceph OSD (3 Items)			
Number of OSDs	2016-01-16 18:17:16	7	-
Number of OSDs up	2016-01-16 18:17:18	7	-
Number of OSDs in	2016-01-16 18:17:21	7	-
Ceph storage (8 Items)			
Ceph pgs	2016-01-16 18:17:06	528	-
Ceph pools	2016-01-16 18:17:08	14	-
Ceph data stored	2016-01-16 18:17:11	62.17 GB	-
Ceph objects stored	2016-01-16 18:17:12	37011	-
Ceph pgmap version	2016-01-16 18:17:14	151491	+2
Ceph storage available	2016-01-16 18:17:50	5.14 TB	-
Ceph storage total	2016-01-16 18:17:51	5.34 TB	-
Ceph storage used	2016-01-16 18:17:51	202.69 GB	-

Slika C.5: Prikaz podatkov v nadzornem sistemu Zabbix.

Literatura

[1] S. W. Schlosser in S. Iren, „Database storage management with object-based storage devices“, v *Proceedings of the First International Workshop on Data Management on New Hardware*, Baltimore, 2005.

[2] J. Kanishk, „CSE 598D-Storage Systems Survey, Object-Based Storage“, The Pennsylvania State University, University Park, 2007.

[3] Amazon Web Services, Inc., „Amazon Simple Storage Service (S3) - Object Storage“, 2016. [Elektronski]. Dostopno na: `http://aws.amazon.com/s3/`. [Dostopano 12. 1. 2016].

[4] Amazon Web Services, Inc, Amazon Simple Storage Service Developer Guide API Version 2006-03-01, Seattle: Amazon Web Services, Inc, 2016.

[5] L. Richardson in S. Ruby, RESTful Web Services, Sebastopol: O'Reilly Media Inc., 2007.

[6] GitHub, Inc., „The boto project“, 2016. [Elektronski]. Dostopno na: `https://github.com/boto`. [Dostopano 28. 2. 2016].

[7] B. Callaghan, NFS Illustrated, Reading: Addison Wesley Longman, Inc., 2000.

[8] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh in B. Lyon, „Design and Implementation of the Sun Network Filesystem“, Sun Microsystems, Inc, Mountain View, 1985.

[9] J. Ts, R. Eckstein in D. Collier-Brown, Using Samba, Second Edition, Sebastopol: O'Reilly, 2003.

[10] GitHub, Inc., „s3fs-fuse“, 2016. [Elektronski]. Dostopno na: `https://github.com/s3fs-fuse/s3fs-fuse`. [Dostopano 28. 2. 2016].

[11] A. Rajgarha in A. Gehani, „Performance and Extension of User Space File Systems“, *SAC,* št. 10, 2010.

[12] S. A. Weil, A. W. Leung, S. A. Brandt in C. Maltzahn, „RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters“, v *2nd Parallel Data Storage Workshop*, Reno, 2007.

[13] S. A. Weil, S. A. Brandt, E. L. Miller in C. Maltzahn, „CRUSH: Controlled, Scalable, Decentrilized Placement of Replicated Data“, v *SC2006*, Tampa, 2006.

[14] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long in C. Maltzahn, „Ceph: A Scalable, High-Performance Distributed File System“, v *OSDI '06: 7th USENIX Symposium on Operating Systems Design and Implementation*, Santa Cruz, 2006.

[15] Inktank Storage, Inc., „Monitor Config Reference“, 2014. [Elektronski]. Dostopno na: `http://docs.ceph.com/docs/hammer/rados/configuration/mon-config-ref/`. [Dostopano 23. 2. 2016].

[16] L. Lamport, „The Part-Time Parliament“, *ACM Transactions on Computer Systems,* Izv. 16, št. 2, pp. 133-169, 1998.

[17] S. A. Weil, K. T. Pollack, S. A. Brandt in E. L. Miller, „Dynamic Metadata Management for Petabyte-scale File Systems“, v *IEEE*, Santa Cruz, 2004.

[18] T. M. Jones, „Ceph: A Linux petabyte-scale distributed file system“, 4. 6. 2010. [Elektronski]. Dostopno na: `http://www.ibm.com/developerworks/library/l-ceph/`. [Dostopano 28. 2. 2016].

[19] M. Ludvig, „Amazon S3 Tools: Command Line S3 Client Software and S3 Backup“, 20. 1. 2016. [Elektronski]. Dostopno na: `http://www.s3tools.org/s3cmd`. [Dostopano 27. 2. 2016].

[20] The Apache Software Foundation, „The Apache HTTP Server Project,“ 2015. [Elektronski]. Dostopno na: `http://httpd.apache.org/`. [Dostopano 28. 2. 2016].

[21] Inktank Storage, Inc., „Configuring Ceph object gateway“, 2014. [Elektronski]. Dostopno na: `http://docs.ceph.com/docs/master/radosgw/config/`. [Dostopano 28. 2. 2016].

[22] Python Software Foundation, „Download Python“, 2016. [Elektronski]. Dostopno na: `https://www.python.org/downloads/`. [Dostopano 28. 2. 2016].

[23] Zabbix LLC., „Zabbix: The Enterprise-class Monitoring Solution for Everyone“, 2016. [Elektronski]. Dostopno na: `http://www.zabbix.com/download.php`. [Dostopano 28. 2. 2016].

[24] J. Tomšič, „Delovna praksa: Zaključno poročilo“, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Ljubljana, 2014.

[25] Zabbix SIA., „User Parameters (Zabbix Documentation 2.4)“, 2015. [Elektronski]. Dostopno na: https://www.zabbix.com/documentation/2.4/manual/config/items/userparameters. [Dostopano 28. 2. 2016].